Hyperparameter Optimization in Transfer Learning for Improved Pathogen and Abiotic Plant Disease Classification

Asha Rani K P*, Gowrishankar S
Department of Computer Science and Engineering, Dr. Ambedkar Institute of Technology,
Bengaluru – 560056, Karnataka, India, Affiliated to VTU, Belagavi – 590018, Karnataka, India

Abstract—**The application of machine learning, particularly through image-based analysis using computer vision techniques, has greatly improved the management of crop diseases in agriculture. This study explores the use of transfer learning to classify both spreadable and non-spreadable diseases affecting soybean, lettuce, and banana plants, with a special focus on various parts of the banana plant. In this research, 11 different transfer learning models were evaluated in Keras, with hyperparameters such as optimizers fine-tuned and models retrained to boost disease classification accuracy. Results showed enhanced detection capabilities, especially in models like VGG_19 and Xception, when optimized. The study also proposes a new approach by integrating an EfficientNetV2-style architecture with a custom-designed activation function and optimizer to improve model efficiency and accuracy. The custom activation function combines the advantages of ReLU and Tanh to optimize learning, while the hybrid optimizer merges feature of Adam and Stochastic Gradient Descent (SGD) to balance adaptive learning rates and generalization. This innovative approach achieved outstanding results, with an accuracy of 99.96% and an F1 score of 0.99 in distinguishing spreadable and non-spreadable plant diseases. The combination of these advanced methods marks a significant step forward in the use of machine learning for agricultural challenges, demonstrating the potential of customized neural network architectures and optimization strategies for accurate plant disease classification.**

Keywords—Spreadable diseases; non-spreadable diseases; transfer learning; Keras; optimizers; CNN; underfitting and overfitting; retraining the models; base models; finetuning; abiotic; biotic; infectious and non-infectious diseases; custom optimization techniques; hyperparameter tuning in neural networks; hybrid activation functions

I. Introduction

Over 80,000 plant diseases are known to exist in the world. Crop diseases often harm crop plants, which can result in major economic and agricultural losses [1]. If a plant disease is induced by an environmental element and is not spread from one plant to another, it is referred to be abiotic, or non-infectious. Diseases classified as biotic or infectious are those brought on by pathogens like viruses, fungi and nematodes.

Pathogens that affect animals as well as humans mimic plant illnesses. Fungus, organisms that mimic fungal, bacterial, phytoplasmas, viral, viral vectors, nematodes and parasitic higher plants are examples of plant diseases.

In order to establish and maintain food security and revenue sources for a developing world, it is more crucial to safeguard plants against diseases. Severity of plant diseases can be reduced with the aid of early identification.

The first lettuce farms were established in ancient Egypt as it was the most widely consumed salad produce and has substantial economic worth. Lettuce is more susceptible to biotic than abiotic illnesses. The study highlights crucial and substantial diseases, such as downy mildew, which may spread swiftly to impact most plants in a crop. Lettuce diseases can cause significant damage and occasionally full crop loss [2]. Some diseases, such as downy and powdery mildews, can spread swiftly and harm the majority of the plants in a crop.

Diseases and insect pests are the main issues in soybean production. To get a broader perspective on spreadable and non-spreadable diseases, soybean plant is chosen. This calls for careful diagnosis and prompt handling to prevent the soybean crops from suffering significant losses. The world's soybean production is projected to be 333.67 million tonnes in 2019–2020 from a total area of 120.50 million hectares [3].

It is crucial to learn more about the spreadable and non-spreadable diseases that affect various plant parts, including the leaves, fruits, stems, nodes and roots. The banana crop fits best into this category because each part of the plant has a variety of uses, including medicinal properties for the stem and roots and maintaining the health of the soil. Additionally, with an output of 97.5 million tonnes, bananas are a significant fruit crop on a worldwide scale. It has a total yearly output of 490,710,000 hectares producing 16.91 million tonnes [4].

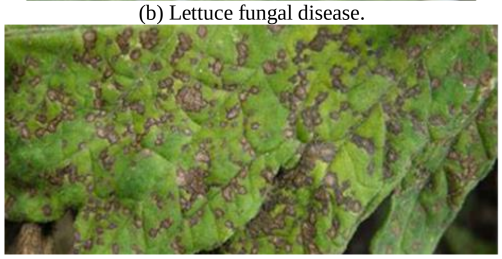
Bananas are an important fruit crop in India.They are a staple food for many people in the country and are also widely used in various dishes. Panama disease, also known as Fusarium wilt, is a serious threat to banana production worldwide. It is caused by a fungus that infects the root system of the banana plant, ultimately causing the plant to wilt and die. Aphids are a common pest that can also infect banana plants. They feed on the sap of the plant, which can weaken it and make it more susceptible to other pests and diseases. It is important for farmers to monitor their banana crops closely and take steps to prevent and control these pests and diseases to protect their yield.

Food security is at risk from plant diseases because they can harm crops, lowering food production and driving up food prices. Leaf blight, septoria blight, powdery mildew and downy mildew, which can be fungal, are the main diseases impacting the lettuce crop. Bacterial rust and downy mildew are the diseases that affect soybeans, whereas weevil, soft rot, aphids, and few may affect bananas [5]. Fig. 1 and 2 shows the illnesses of lettuce and soybean.

Deep learning is a cutting-edge technique for object recognition and image processing that improves categorisation of numerous crop diseases [6]. One well-liked method in deep learning where pre-trained models are modified to perform a new job is transfer learning. Deep Transfer Learning (DTL) creates a applied novel framework for predictive analytics and digital image processing that is more accurate and has enormous potential for crop disease identification. A potential method for recognising diseases onsite is the DTL technique, which also offers a quick way to adapt created models to the constraints imposed by mobile applications [7]. This would be very useful in a real-world field scenario.

(a) Lettuce viral disease.

(b) Lettuce fungal disease.


(c)Lettuce Bacterial Disease.

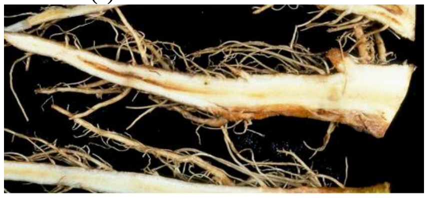
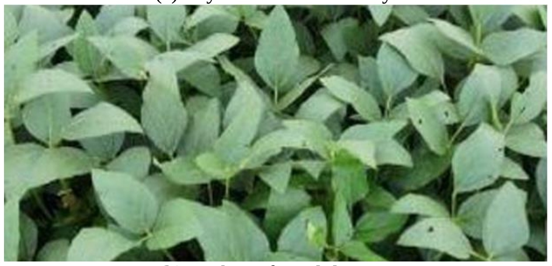

(d) Lettuce Non-Infectious Disease (Salt burn).

Fig. 1. Some major plant diseases found in lettuce plant dataset.

A variety of factors, including that of the high-definition camera, high efficient processing and many built-in accessories, enable automatic disease identification. The accuracy of the outcomes has increased because of the use of cutting-edge techniques like deep learning and machine learning. Our experimental results represent significant advances in the understanding of the severity of plant diseases. The paper is organised as follows for the following sections: Section I Introduction, Section II Literature Review, Section III highlights Methodology that includes expanded dataset description, Augmentation and Activation functions, Section IV describes Performance evaluation. Performance reviews go into great detail, Section V is concerned with implementation, results are providedd in tabular format and Section VI acts as a conclusion.

(a) Soybean Iron deficiency.


(b) Soybean fungal disease.

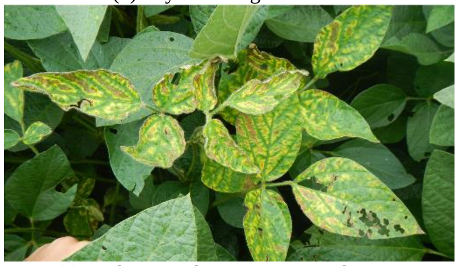

(c) Soybean Diabrotica speciosa disease.

Fig. 2. Some major plant diseases found in soybean plant dataset.

II. Literature Survey

N. Saranya et al. [8] have categorized many ailments that affect the leaves and fruits of the banana plant. Fuzzy c-means, histogram-based equalization and artificial neural networks all have important roles in the proposed approach. The image is divided using fuzzy c-means and the histogram is then utilized to transform it without losing any of the details of the banana plant. In this study, a better categorization strategy is recommended in order to deliver the best return.

Michael Gomez Selvaraj et al. [9] applied pixel-based banana classification using the Random Forest (RF) model utilizing integrated features of Vegetative Indices (VI) and Principal Component Analysis (PCA) to map banana under mixed-complex African settings. Gomez Selvaraj et al. provided higher & low-resolution aerial (UAV and satellite) photos with cutting-edge computer vision algorithms to achieve more than 90% accuracy under actual settings (Smart phone-based AI applications).

A high-definition camera is used to take photos of the early and intermediate phases of soybean disease and the photos are then expertly batched into uniform sizes. The picture segmentation methods used by E. Miao et al. [10] include lab grayscale map, ultragreen feature approach, genetic algorithm and threshold segmentation. Next, the results are filtered using the median and corroded expansion. A Convolutional Neural Network (CNN) that uses a MultiLayer Perceptron (MLP) framework to execute supervised learning of the network and achieves an average recognition rate of 94.87% is seen in the soybean illness picture identification experiment.

Three disease groups of soybean leaves were examined by Sachin B. Jadhav et al. [11] bacterial blight, frogeye leaf spot, and septoria brown spot. The diseased leaf area is segmented using incremental K-means clustering. Color and texture data are recovered using the R, G, B color space and the Gray Level Co-occurrence Matrix (GLCM), respectively. SVM and K-Nearest Neighbors Algorithm (KNN) are used in a classification technique to identify the exact kind of leaf disease. The results demonstrate that the SVM classifier approach outperforms the KNN methodology with efficiencies of 87.3% and 83.4%, respectively.

Elham Khalili et al. [12] examined and compared six ML methods for identifying the ailment known as charcoal rot in their study that was published in science. Healthy plants were gathered from the stem and root of soybean plants during the ripening stage based on the maturity's symptomless qualities. R7(Yellowing of the leaves and yellow pods at 50% growing stage) was chosen for sick plants based on physical criteria indicates the presence of bright grey and mycelium on the root and stem. Gradient Tree Boosting and Support Vector Machines performed better than Regularized Logistic Regression, MultiLayer Perceptron and Random Forest techniques.

Miao Yu et al. [13] used the OTSU technique, which decreases the effect of the background on the disease images. Using ResNet18 and RANet, the model's effectiveness in the test set was confirmed and assessed. The response time was 0.0514 seconds, the F1-value was 98.52, and the RANet average recognition rate was 98.49%. Compared to ResNet18, the identification rate increased by 1.15 percent, the F1-value increased by 1.17 and 0.0133 seconds were saved while identifying illnesses from images.

Lack of calcium makes tip-burn, which is common in lettuce plants cultivated indoors, worse. Photos of tip-burn lettuce were illuminated using white, red and blue LEDs, and these images served as the training, validation and testing datasets for a deep-learning detection method. The detection approach developed by Munirah Hayati Hamidon et al. [14] was based on three detectors: CenterNet, YOLOv4 and YOLOv5. YOLOv5 beat the other two models tested, with an accuracy of 84.1% mAP.

Positive and negative samples from each kind of weed and crop were chosen by Kavir Osorio et al. [15] and taken. There are just a few of weed characteristics that remain consistent, making identification difficult. The identification of the vegetation was done alternatively using multispectral bands. The R-CNN model distinguished itself for its accuracy in detecting the crop and showing the edges, making it a tactic that may be recommended for addressing problems like fruit detection. The RCNN and HOG-SVM-based algorithms were shown to be the most trustworthy using the Bland-Altman approach. The YOLO strategy exaggerates the high levels of cannabis coverage in contrast to the other two.

According to J. Amara et al. [16], who used the LeNet architecture as a Convolutional Neural Network to classify the data, banana leaf disease may now be classified using deep learning. This strategy stabilized after 25 iterations. This research demonstrated its effectiveness in a variety of picture situations, including ones with a complex background and various sizes and orientations.

W. Liao et al. [17] proposed using the SVM classifier in a machine learning-based strategy for early identification of banana disease. Hyperspectral images taken at close range are utilized in this instance. When using spectral and morphological data, the classifiers' outputs have an overall accuracy of 96% for early detection, 90% for mid-detection, and 92% for late detection.

More research is being done to detect and classify the disease, not just for banana leaves but also for the majority of food crops including rice, maize, apple, cheerio and other well-known plants. Here are a few of these judgements.

A superior convolutional neural network should be used to classify apple plant and cherry plant diseases, according to [18].

In the extremely packed growing conditions of indoor settings, early diagnosis of tip-burns in lettuce is vital in order to reduce the cost of human identification and boost lettuce quality and production. Shimamura et al. [19] created a system for tip-burn identification in plant factories by using GoogLeNet to classify two different types of tip-burn from a single picture of lettuce.

The most recent Neuron Compute Stick pretrained Movidius of deep CNN model from Intel provided an accuracy rate of 88.46% for Mishra et al. [20]'s system for classifying and diagnosing maize leaf diseases. The system was implemented on a Raspberry Pi 3.

K-means segmentation and multiclass support vector machines were used by Kumar et al. [21] to identify and classify different plant leaf diseases (SVM-based classification). Compared to other approaches, the detection accuracy is much higher.

To eliminate manual feature stage modelling, Mazzia et al. [22] created an LC&CC deep learning model that blends Recurrent Neural Networks (RNN) with Convolutional Neural Networks (CNN).

Researchers Cetin et al. [23] used six different machine learning algorithms to analyze and classify six sunflower varieties (105 single seeds) based on their fatty acid and mineral composition, biochemical traits and physical characteristics. These algorithms included Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM), Multiple Linear Regression (MLR), Naive Bayes (NB) and MultiLayer Perceptron (MLP).

Sharif et al. [24] proposed a hybrid feature selection method, which included the principal components analysis score, entropy, skewness-based covariance vector and Multiclass-SVM (MSVM), produced true positive rates of 96.9% and

97.1% for the detection of anthracnose disease and melanose disease on citrous leaves.

Machine learning models including K-Nearest Neighbor (KNN), Support Vector Machine (SVM), and other machine learning models have been widely used as classifiers to find anomalous areas on crop leaves. With a detection rate of 90.5%, Lu et al. [25] used Fisher discriminant analysis to identify anthracnose crown rot in the early stages of affected strawberry leaves inside.

To detect early blight on potato leaves, Vijver et al. [26] used partial least squares discriminant and discovered a positive predictive value of 0.92. This study demonstrated that artificial intelligence can accurately identify aberrant leaves in a range of crops. Therefore, using machine learning algorithms to detect yellow and wilted lettuce leaves in hydroponic systems is encouraging.

III. METHODOLOGY

A. Dataset Description

Performance evaluation of existing transfer learning pretrained models for plant disease classification is done using the Lettuce, Soybean and Banana dataset. Non-Spreadable diseases caused by abiotic factors include herbicide injury which turn leaves or leaf veins yellow or red. Calcium strengthens plant cell walls and salt burn is a result of the plant's inability to supply enough calcium for developing leaves during periods of rapid growth. The dataset for the proposed research consists of images of Lettuce plant and Soybean Plant infected by non-spreadable diseases and images affected by spreadable diseases. These images have been obtained from CrowdAI [27] and PlantVillage dataset [28]. The images have been augmented and brought up to 628 images for the former and 1845 images for the latter, as represented in Fig. 3 and 4. Each image maintains a fixed width and height of 256x256 pixels.

Two mobile phones and a UAV were used to take pictures of soybeans. Three groups make up the dataset: (I) photos of healthy plants, (II) pictures of plants harmed by caterpillars, and (III) pictures of plants harmed by Diabrotica speciosa. To meet our demands, the photos have undergone processing and augmentation [29].

The banana plant is vulnerable to bacterial, fungal, and viral diseases that can affect different parts of the plant. For research purposes, the PSFD-Musa dataset [30] was utilized, which contains pre-existing enhanced photos. Fig. 5 to 11 represent the dataset, showcasing the spreadable disease affected regions such as Node, Leaf, Banana, and Fruit.

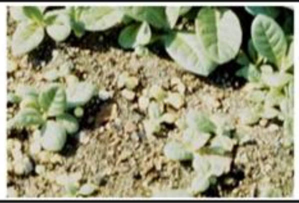
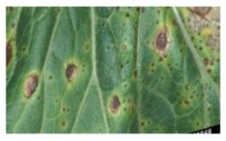
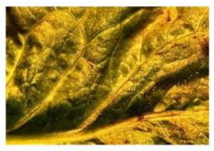
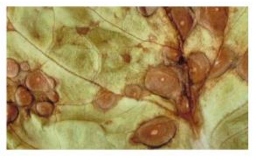
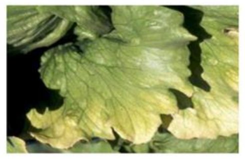
Lettuce Dataset			
Non-Spreadable Disease	Herbicide Injury	Salt Burn	Healthy
No. of Original images	26	12	26
No. of Augmented Images	208	212	208

Fig. 3. Lettuce (Non-Spreadable) Dataset Description.

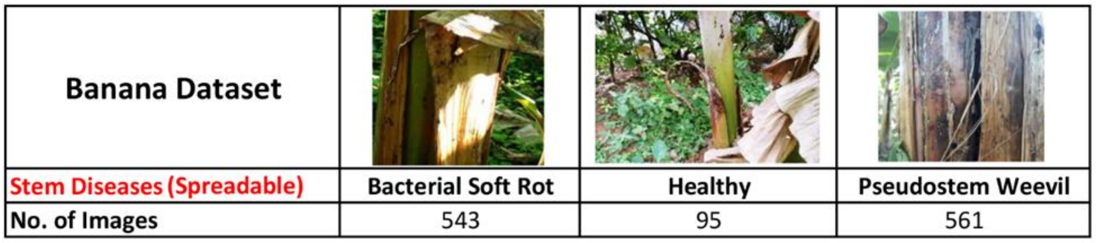
Lettuce Dataset				
Spreadable Disease	Bacterial	Fungal Downy Mildew	Funga Powdery Mildew	Fungal Septoria Blight
No. of Original images	35	32	26	19
No. of Augmented Images	220	261	464	238

(a)

Lettuce Dataset			
Spreadable Disease	Fungal wilt and leaf blight	Viral	Healthy
No. of Original images	10	17	26
No. of Augmented Images	212	232	208

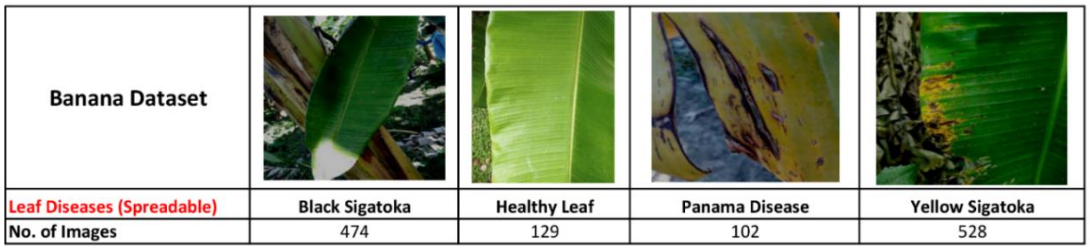
(b)

Fig. 4. Lettuce (Spreadable) Dataset Description.



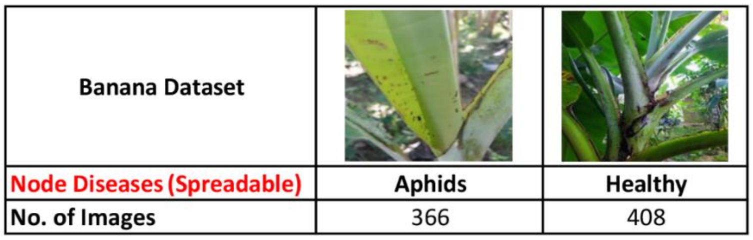
Banana Dataset			
Stem Diseases (Spreadable)	Bacterial Soft Rot	Healthy	Pseudostem Weevil
No. of Images	543	95	561

Fig. 5. Dataset classification of Banana Stem.



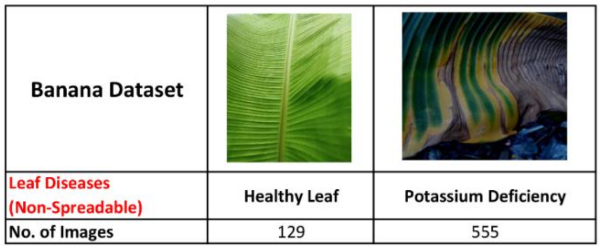
Banana Dataset				
Leaf Diseases (Spreadable)	Black Sigatoka	Healthy Leaf	Panama Disease	Yellow Sigatoka
No. of Images	474	129	102	528

Fig. 6. Dataset classification of Banana Leaf (Spreadable).



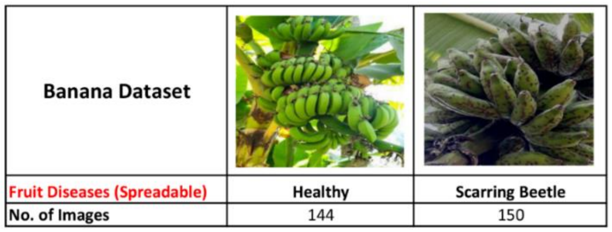
Banana Dataset		
Node Diseases (Spreadable)	Aphids	Healthy
No. of Images	366	408

Fig. 7. Dataset classification of Banana Node



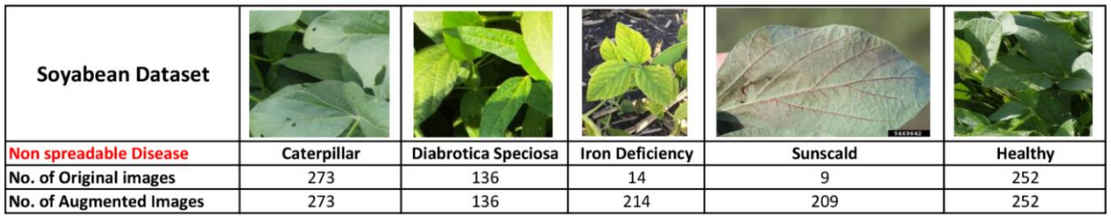
Banana Dataset		
Leaf Diseases (Non-Spreadable)	Healthy Leaf	Potassium Deficiency
No. of Images	129	555

Fig. 8. Dataset classification (non-Spreadable) disease in Banana crop.



Banana Dataset		
Fruit Diseases (Spreadable)	Healthy	Scarring Beetle
No. of Images	144	150

Fig. 9. Dataset classification spreadable disease in Banana Fruit.



Soyabean Dataset					
Non spreadable Disease	Caterpillar	Diabrotica Speciosa	Iron Deficiency	Sunscald	Healthy
No. of Original images	273	136	14	9	252
No. of Augmented Images	273	136	214	209	252

Fig. 10. Dataset classification of Soybean Non-Spreadable Diseases.

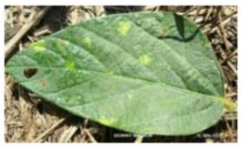
Soyabean Dataset			
Spreadable Disease	Rust	Downy Mildew	Healthy
No. of Original images	61	27	252
No. of Augmented Images	261	227	252

Fig. 11. Dataset classification of Soybean Spreadable Diseases.

The dataset includes images of leaves affected by diseases like Black Sigatoka, Panama, and Yellow Sigatoka (Fig. 5-7), with a total count of 1104. Stem diseases, namely Bacterial Soft Rot and Pseudostem Weevil, also have 1104 corresponding images. The Banana node is affected by Aphids (Fig. 8), and the Banana Fruit is affected by Scarring Beetle (Fig. 9), with 366 and 150 images obtained, respectively.

To enhance the model's efficiency and performance, self-captured healthy banana images were incorporated. Fig. 7 illustrates images of healthy nodes. Additionally, potassium deficiency, caused by abiotic factors, has an adverse impact on banana leaves (Fig. 8). The dataset also includes 150 images of spreadable diseases in banana fruit (Fig. 9).

In addition to the banana dataset, 740 photos of soybean plants infected with spreadable diseases (Fig. 10) and 1084 non-spreadable images (Fig. 11) were obtained for comparative analysis.

B. Augmentation

Convolutional Neural Networks (CNN) are widely used for image classification. However, the quality and quantity of training data significantly impact model performance. Insufficient or imbalanced data can lead to poor generalization. Techniques like oversampling or undersampling can address class imbalance [31]. Data augmentation, including affine transformations and color manipulation, is a popular method to increase dataset size. Classical approaches may not always improve accuracy or address overfitting effectively. Affine transformations include rotation, reflection, scaling, and shearing. Additional techniques like permutation rotate, random zoom, variation in shear, random crop, and flip can be used. After data augmentation and balancing, the dataset consisted of 200 augmented images per class [32].

C. Activation Function

An activation function is a mathematical function that is applied to the input of a neural network node or a layer of nodes. The activation function is used to introduce non-linearity into the network, which is necessary for the network to learn complex patterns in the input data. Without activation functions, a neural network would essentially be a linear model, which is limited in its ability to learn complex relationships.

Activation methods that are frequently employed based on a few desirable characteristics include:

1) Nonlinear: Whenever the activation function is nonlinear, it has been shown that a two-layer neural network is an excellent approximator of any function. The identical activation function does not satisfy this condition. When many layers employ the same activation function, the network as a whole is equivalent to a single-layer model.

2) Range: Gradient-based training techniques have a tendency to be more stable when the activation function's range is finite, since only a small number of weights are significantly affected by pattern presentations. Since most of the weights are strongly affected by pattern presentations when the range is unlimited, training is often more effective. Short learning rates are often required in the latter scenario.

3) Continuously differentiable: For the purpose of allowing gradient-based optimization approaches, this property is desirable (ReLU is not continuous differentiable and has some challenges with it, but it is still achievable). Because the binary step activation function is not differentiable at zero and differentiates to zero for all future values, gradient-based techniques cannot advance with it.

4) Monotonic: A single-layer model's related error surface is always guaranteed to be convex when the activation function is monotonic.

5) Approximates near the origin: When activation functions have this property, the neural network can learn efficiently when its weights are initialized with low-level random values. If the activation function differs from identity near to the origin while initializing the weights, more care must be taken.

Each activation function has advantages and disadvantages, so we must be cautious when choosing one. Following are some frequent considerations to make while selecting an activation function:

1) When it comes to classification issues, sigmoid [33] functions (including softmax) and their combinations often perform better.

2) Due to the vanishing gradient issue, sigmoid and tanh functions continue to be avoided in hidden layers.

3) Tanh is typically avoided because of the dead neuron issue [34].

4) Because it produces superior results, ReLU activation function is frequently employed and is the default option (than sigmoid and tanh) [35].

5) However, the ReLU function should only be utilized in the buried layers (and not in the output layer).

6) In cases of regression issues, an output layer's activation function can be linear, however nonlinear activation functions are required for classification tasks.

7) The leaky ReLU function is the ideal option if we come into an instance of dead neurons in our networks.

8) For any kind of neural network, the ReLU activation function is presently the one that is most frequently employed for the hidden layers (but never for the output layer).

9) Swish activation should only be utilized for bigger neural networks with depths of more than 50 layers, even though it does not consistently beat ReLU in complicated applications [36].

10) The output (top-most) layer should be triggered by the sigmoid function for 2-class applications, as well as for multi-label classification.

11) The output layer must be triggered using the softmax activation function for multi-class applications.

12) A basic regression neural network should just employ the linear activation function in the output layer.

13) The tanh activation function is recommended for the hidden layer in Recurrent Neural Networks (RNN). By default, it is configured by TensorFlow.

14) In some circumstances, switching to a leaky ReLU might produce better outcomes and overall performance if ReLU is unable to deliver the desired results.

Some of the known Activation functions are:

1) The sigmoid function: Logistic regression and simple neural network implementation both use sigmoid functions. The fundamental activation units in machine learning are sigmoid functions. However, because of a number of limitations, it is simply not advisable to use complicated neural network sigmoid functions (vanishing gradient problem). Given that it is among the most frequently used activation functions, it serves as an excellent introduction for those who are naïve to data science and machine learning. Whilst the sigmoid function and its derivative are simple to use and help reduce the time required to develop models, there is a considerable downside of data lost since the derivative has a constrained range.

2) Tanh function: The tanh function partially addresses the drawback of the sigmoid function. Its key feature is that its curve is symmetric across the origin and has coefficients that range from -1 to 1 [34]. This does not, however, mean that the fading or bursting gradient problem does not occur. It does exist for tanh, however unlike Sigmoid, it is centered at zero, making it more ideal than Sigmoid Function.

3) ReLU (Rectified Linear Units) and Leaky ReLU: ReLU functions, as opposed to Logistic Activation functions, are currently used in the majority of Deep Learning applications, such as computer vision, natural language processing, speech recognition, deep neural networks, etc [35]. ReLU outperforms tanh or sigmoid functions in terms of application-level manifold convergence speed. Among the ReLU variations are Leaky ReLU, Parametric ReLU, Parametric Softplus (SmoothReLU), Noisy ReLU, and ExponentialReLU (ELU) [36].

4) Softmax function: The Softmax activation function which not only turns our output into a [0, 1] range but also changes each outcome so that the sum of each is 1 [37], is extremely fascinating. Softmax produces probability distribution as a result. In logistic regression model (multivariate), Softmax is used for multi-classification while Sigmoid is employed for binary classification.

D. Mathematical Approach for the Considered Procedure

The field of machine learning relies heavily on mathematical principles and techniques to design, train, and optimize models that can make predictions or learn patterns from data. This mathematical approach enables us to create powerful algorithms capable of solving a wide range of tasks, from image recognition and natural language processing to financial predictions and recommendation systems.

At the core of the mathematical approach in machine learning is the idea of formulating the learning problem as an optimization task. The goal is to find the model's parameters that minimize a certain objective function, such as the mean squared error in regression tasks or the cross-entropy loss in classification tasks. This process involves using various mathematical tools to represent the model, compute gradients, and iteratively update the parameters to approach the optimal solution. let's go through each of these layers, providing a brief introduction and their mathematical formulas:

1) Convolutional Layer (Conv layer): Convolutional layers are the fundamental building blocks of Convolutional Neural Networks (CNNs). They are designed to automatically and adaptively learn spatial hierarchies of features from input data such as images. A convolutional layer applies convolutional operations to input data using learnable filters (kernels) to detect local patterns and features.

The output of a convolutional layer can be represented as follows:

Given an input feature map X with dimensions (height, width, channels), and a set of learnable filters W of size (filter_height, filter_width, input_channels, output_channels), the convolution operation can be represented as:

$$\mathrm{Y}[\mathrm{i},\mathrm{j},\mathrm{k}] = \Sigma\,\Sigma\,\Sigma\,\mathrm{X}[\mathrm{p}+\mathrm{i},\mathrm{q}+\mathrm{j},\mathrm{r}] * \mathrm{W}[\mathrm{p},\mathrm{q},\mathrm{r},\mathrm{k}]$$

Here,

- Y[i, j, k] is the value of the output feature map at position (i, j) in the k-th channel.
- X[p+i, q+j, r] is the value of the input feature map at position (p+i, q+j) in the r-th channel.
- W[p, q, r, k] is the value of the learnable filter at position (p, q) in the r-th input channel and k-th output channel.
- The summation is performed over all spatial positions (p, q) of the filter and all input channels (r).

2) MaxPooling Layer (MaxPooling): MaxPooling is a downsampling technique commonly used in CNNs to reduce the spatial dimensions of the feature maps while retaining the most important information. It works by dividing the input feature map into non-overlapping regions and taking the maximum value within each region.

The output of a MaxPooling layer can be represented as follows:

Given an input feature map X with dimensions (height, width, channels), and a pooling window of size (pool_height, pool_width), the MaxPooling operation can be represented as:

$$Y[i,j,k] = max\left(X\left[i * pool_{height} : (i+1) * pool_{height}, j * pool_{width} : (j+1) * pool_{width}, k\right]\right)$$

Here,

- Y[i, j, k] is the value of the output feature map at position (i, j) in the k-th channel.
- The max function takes the maximum value within the pooling window.

3) SeparableConv Layer (Depthwise Separable Convolution): The SeparableConv layer is an alternative to standard convolutions designed to reduce computation and model size while maintaining representational capacity. It splits the convolution operation into two steps: depthwise convolution and pointwise convolution.

The output of a SeparableConv layer can be represented as follows:

Given an input feature map X with dimensions (height, width, channels), a depthwise kernel DW of size (filter_height, filter_width, channels), and a pointwise kernel PW of size (1, 1, channels, output_channels), the SeparableConv operation can be represented as:

$$Y[i,j,k] = \Sigma\, \Sigma\, X[i+p, j+q, r] * DW[p,q,r] * PW[1,1,r,k]$$

Here,

- Y[i, j, k] is the value of the output feature map at position (i, j) in the k-th channel.
- X[i+p, j+q, r] is the value of the input feature map at position (i+p, j+q) in the r-th channel.
- DW[p, q, r] is the value of the depthwise kernel at position (p, q) in the r-th channel.
- PW[1, 1, r, k] is the value of the pointwise kernel at position (1, 1) in the r-th input channel and k-th output channel.
- The summation is performed over all spatial positions (p, q) of the depthwise kernel and all input channels (r).

4) GlobalAveragePooling2D Layer: Global Average Pooling 2D is another downsampling technique used in CNNs, often as an alternative to fully connected layers at the end of the network. It computes the average value of each channel of the feature map, reducing the spatial dimensions to a single value per channel.

The output of a GlobalAveragePooling2D layer can be represented as follows:

Given an input feature map X with dimensions (height, width, channels), the Global Average Pooling operation can be represented as:

$$Y[k] = \left(\frac{1}{(height * width)}\right) * \Sigma\, \Sigma\, X[i,j,k]$$

Here,

- Y[k] is the value of the output for the k-th channel.
- The summation is performed over all spatial positions (i, j) of the feature map.

5) Dense Layer (Fully Connected Layer): The Dense layer is the standard fully connected layer in neural networks. It connects every neuron from the previous layer to every neuron in the current layer. The Dense layer performs a linear transformation followed by an activation function.

The output of a Dense layer can be represented as follows:

Given an input vector X of size (input_units) and the weight matrix W of size (input_units, output_units), the Dense layer operation can be represented as:

$$Y = activation\_function(X * W + b)$$

Here,

- Y is the output vector.
- activation_function is the non-linear activation function applied element-wise to the linear transformation.
- b is the bias vector of size (output_units).

6) BatchNormalization layer: BatchNormalization is a normalization technique applied to intermediate layers in neural networks to stabilize and accelerate training. It normalizes the activations of each layer's mini-batch, making the network more robust and less sensitive to the scale of the input.

The output of a BatchNormalization layer can be represented as follows:

Given an input feature map X with dimensions (batch_size, features), and learnable scaling and shifting parameters γ and β, the BatchNormalization operation can be represented as:

$$\mu = \frac{1}{batch_{size}} * \Sigma\, X$$

$$\sigma^2 = \frac{1}{batch_{size}} * \Sigma\, (X - \mu)^2$$

$$X_{normalized} = \frac{(X - \mu)}{\sqrt{\sigma^2 + \varepsilon}}$$

$$Y = \gamma * X\_normalized + \beta$$

Here,

- μ and σ² are the mean and variance of the mini-batch.

- X_normalized is the normalized input.
- γ and β are learnable scaling and shifting parameters, respectively.
- ε is a small constant (usually a small value like 1e-5) added for numerical stability.

7) Flatten layer: The Flatten layer is used to reshape the high-dimensional feature maps into a 1D vector, which is then fed into a Dense (fully connected) layer for further processing.

Let's consider an input tensor X with dimensions (batch_size, height, width, channels), where:

batch_size: The number of samples in the batch.

height: The height dimension of the feature maps.

width: The width dimension of the feature maps.

channels: The number of channels (depth) of the feature maps.

The Flatten layer reshapes the input tensor X into a 1D vector with size (batch_size, height * width * channels). This is achieved by simply concatenating all the elements of each feature map in X into a single long vector.

$$Flatten\_output = Reshape(X, (batch\_size, height * width * channels))$$

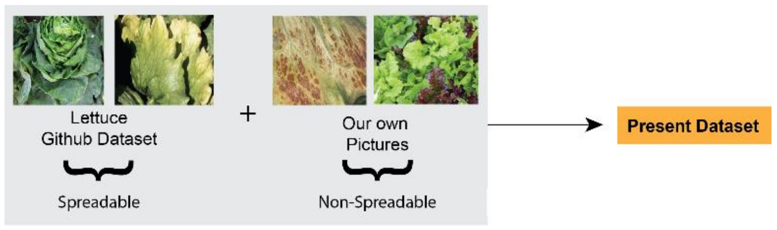
Here, Reshape(X, (batch_size, height * width * channels)) represents the operation of reshaping the input tensor X into the specified dimensions.

IV. Performance Evaluation and Results

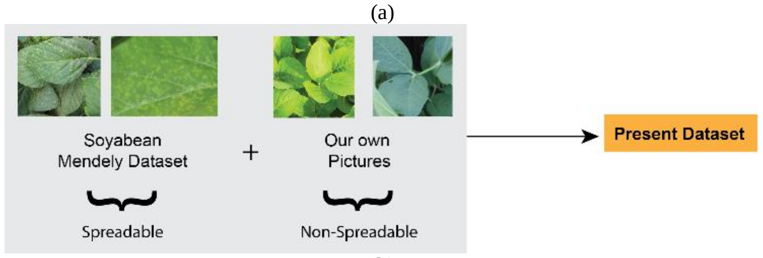
A. Model and Dataset Selection

On datasets for lettuce, soybeans and bananas, 11 Transfer Learning models have been used to identify and categorize disease occurrence. The dataset that was collected from web sources are treated as raw data and organized as indicated in Fig. 12 and 13.

The original dataset was reshuffled, and the resulting dataset is used to train the transfer learning models in Keras module. Divided 38 TL Keras models into 11 different groups, grouped them as families, and primarily selected a member from each group for further study.



(a)



(b)

Fig. 12. (a) Modification of Lettuce Dataset, (b) Modification of Soybean Dataset.

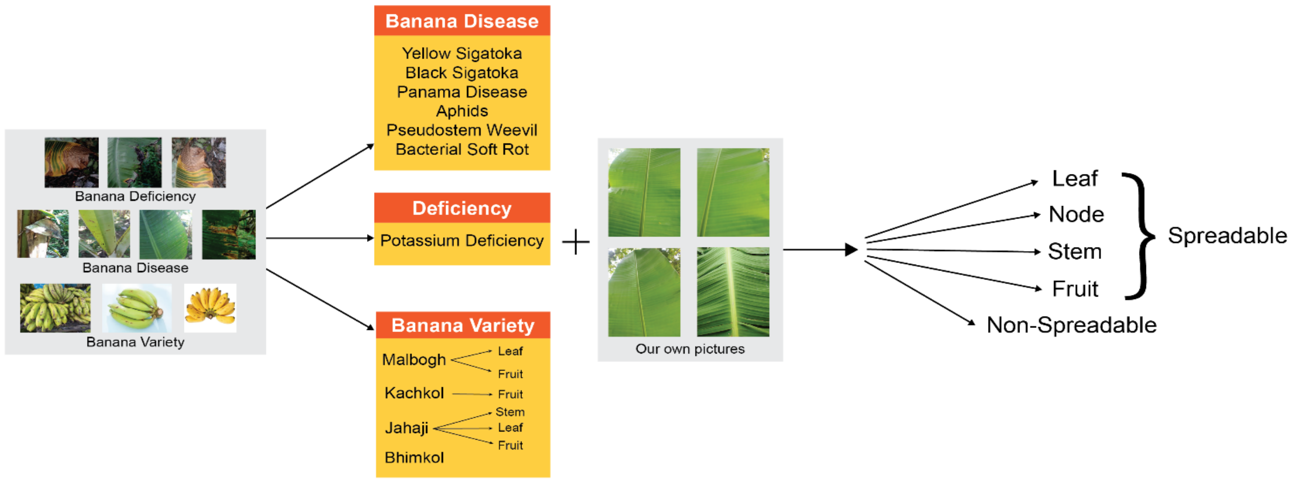


Fig. 13. Variety of banana dataset considered.

The bold models in Table I are chosen for research.

TABLE I. MODEL CLASSIFICATION ACCORDING TO MODELS FAMILY CONCEPT

Family	Model
Xception family	**Xception**
VGG Family	VGG16
	VGG19
ResNet Family	ResNet50
	ResNet50V2
	ResNet101
	ResNet101V2
	ResNet152
	ResNet152V2
Inception Family	**InceptionV3**
	InceptionResNetV2
MobileNet Family	MobileNet
	MobileNetV2
DenseNet Family	DenseNet121
	DenseNet169
	DenseNet201
NASNet Family	**NASNetMobile**
	NASNetLarge
EfficientNet Family	**EfficientNetB0**
	EfficientNetB1
	EfficientNetB2
	EfficientNetB3
	EfficientNetB4
	EfficientNetB5
	EfficientNetB6
	EfficientNetB7
EfficientNetV2 Family	EfficientNetV2B0
	EfficientNetV2B1
	EfficientNetV2B2
	EfficientNetV2B3
	EfficientNetV2S
	EfficientNetV2M
	EfficientNetV2L
ConvNext Family	ConvNeXtTiny
	ConvNeXtSmall
	ConvNeXtBase
	ConvNeXtLarge
	ConvNeXtXLarge

To retrain a transfer learning model, you will need to follow these steps:

1) Choose a pre-trained model: Start by choosing a pre-trained model that you want to use as the base for your model. There are many pre-trained models available in various libraries and frameworks, such as TensorFlow, PyTorch and Keras.

2) Freeze the base model: The pre-trained model will likely have many layers, and you will want to "freeze" the weights of these layers so that they are not updated during training. This will allow you to take advantage of the knowledge learned by the pre-trained model on a large dataset, while still training a new model that is customized for your specific task.

3) Add new layers: Next, you will want to add one or more layers to the model that you can train specifically for your task. These layers should be added on top of the frozen base model.

4) Train the model: Once you have added your new layers, you can compile and train your model using your own dataset. This will allow the model to learn task-specific features that are relevant to your problem.

5) Fine-tune the model: After training, you may want to fine-tune your model by unfreezing some of the layers in the base model and training them along with the new layers. This can help to further improve the performance of your model.

6) Evaluate the model: Once the model has been trained and fine-tuned, it is important to evaluate its performance on a validation set to ensure that it is not overfitting to the training data. You can use metrics such as accuracy, precision, recall and F1 score to evaluate the performance of your model.

7) Tune hyperparameters: You may need to tune hyperparameters such as learning rate, batch size, and number of epochs to optimize the performance of your model. This can be done using techniques such as grid search or random search.

8) Deploy the model: Finally, once the model has been trained and evaluated, it can be deployed in a production environment to make predictions on new, unseen data. This can be done using various deployment strategies such as containerization or serverless functions.

Maintaining a modest learning rate during fine-tuning is an important strategy to avoid over and under-distorting the CNN weights. The learning rate determines the step size at each iteration during the optimization process and a high learning rate can result in large weight updates that may cause the weights to diverge or oscillate. On the other hand, a low learning rate may result in slow convergence or getting stuck in local minima. A modest learning rate strikes a balance between these two extremes, allowing the model to converge towards an optimal solution without over-distorting the weights. The identification of both communicable and non-communicable diseases is done using Transfer learning techniques as shown in Fig. 14.

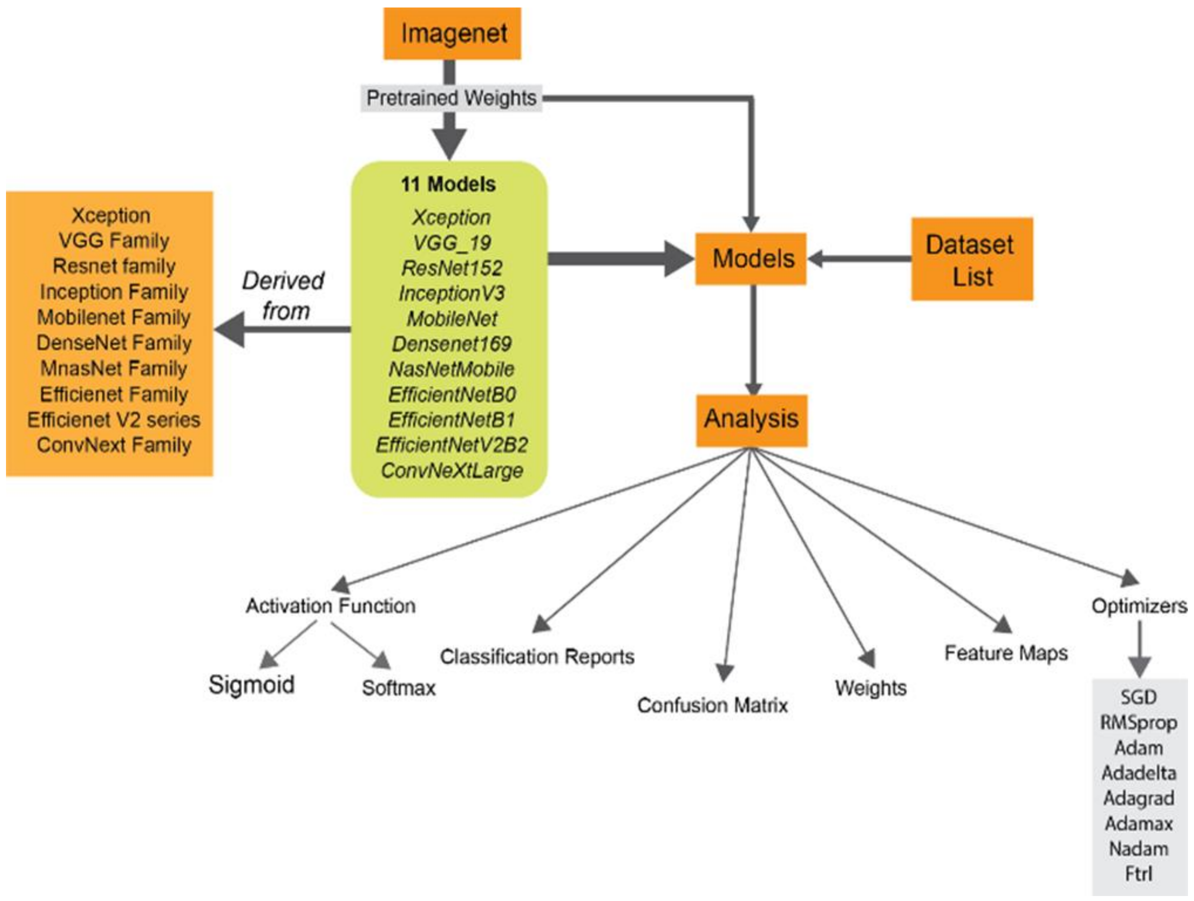


Fig. 14. Approach and analysis of the Transfer Learning Models for plant disease detection in Lettuce, Soybean and Banana.

V. Implementation and Results

The accuracy of 11 TL models selected from Table II, as well as average F1 score for each class belonging to both spreadable and non-spreadable kinds, are shown in Table III. Table IV considers the average F1 score and accuracies for the Lettuce dataset, which contains 7 classes for infectious diseases.

Accuracy is employed when True Positives as well as True Negatives are more necessary, but F1-score is employed when False Negatives but also False Positives are essential.

While F1-score is a superior measure when there are unbalanced classes, as in the example above, accuracy may be utilized when the class distribution is similar. Due to the uneven class distribution that characterizes the majority of real-world classification tasks, F1-score is a superior statistic to use when assessing the model.

A. *What is ConvNext?*

The science of computer vision has long employed residual networks like ResNets. Because of its smaller Residual Block design, it is considerably simpler to train deep neural networks employing skip connections. ResNet will serve as the beginning point because of their incredible accomplishment. The network will be gradually improved from this starting point, and after each enhancement, its performance will be assessed using the dataset and compared to vision transformers.

1) ConvNext: A ConvNet that outperforms Vision Transformers in terms of accuracy, performance and scalability while having the structural simplicity of Convolutional Neural Networks.

B. *Assessment of ConvNext*

In comparison to its vision transformer contemporaries, the new ConvNet, termed ConvNeXt, is not only more accurate but also more scalable. The graph of Fig. 15 compares ConvNext models to their equivalent vision transformers in ImageNet-1K [38].

The Table V displays the accuracy of the 11 models that were trained on the Banana dataset for all 5 classes. The ConvNeXtXLarge model performs best by yielding the most accurate findings, but the baseline model, NASNetMobile, is inappropriate for datasets based on plants, as can be seen in Table V.

The outcomes of Transfer Learning models developed for three separate datasets—lettuce, soybean and banana under various categorizations, infectious and non-infectious illnesses, were covered in the section above. The behavior of the models trained on the same three datasets but combined is covered.

TABLE II. MODEL DESCRIPTION

Model	**Description**
Xception	It makes use of the Xception 71-layer deep convolutional neural network. More than one million pictures from the Imagenet dataset may be used to preload a network that has previously been pretrained. The pretrained network will categorize photos into far more than a thousand more object categories in addition to keyboards, mice, pencils, and other animals.
VGG_19	The total number of layers in the convolutional neural network VGG-19 is 19. More than one million pictures from ImageNet database may be used to preload a network which has previously been pretrained. The pretrained network will categorize photos into far more than a thousand more object categories in addition to keyboards, mice, pencils and other animals.
ResNet152	Detailed Retention Learning, recognizing images with ResNet-152. The bottleneck in TorchVision occurs at the second 3x3 convolution, as opposed to initial 1x1 convolution in the original work. ResNet V1 is the modification that improves accuracy.
InceptionV3	On ImageNet dataset, it has been demonstrated that InceptionV3 image recognition model achieves greater than 78.1% accuracy.
MobileNet	The MobileNet model is a network model that uses depthwise separable convolution as its fundamental unit. It has two layers in its depthwise separable convolution: depthwise convolution and point convolution.
Densenet169	The suggested model includes 4 convolutional layers, 2 maxpool layers, 1 fully connected layer, and three dense layers.
NasNetMobile	More than a million photos from the ImageNet collection were used to train the NASNet-Mobile convolutional neural network. There are more than 1000 different object categories which the network can identify in images, including keyboards, mouse, pens and other animals.
EfficientNetB0	The architecture EfficientNetB0 is launched. The output of this function is a Keras image classification model that can be trained using weights from ImageNet.
EfficientNetB1	The CNN construction and scaling approach EfficientNetB1 equally scales all depths, width and resolution parameters using a compound coefficient.
EfficientNetV2B2	It is a brand-new class of convolutional networks that train faster and more efficiently than older models. We develop this family of models by combining training-aware neural architecture search with scaling to jointly improve training speed and parameter efficiency. A search region that had been widened to include fresh processes like Fused-MBConv was utilized to hunt up the models. Our testing show that EfficientNetV2 models train up to 6.8 times faster than state-of-the-art models despite being much smaller.
ConvNeXtXLarge	ConvNeXT, is said to exceed Vision Transformers in terms of performance (ConvNet).

TABLE III. F1 SCORE AND ACCURACIES OF SOYBEAN DATASETS

Models	**F1 Score (Non-Spreadable)**	**Accuracy (Non-Spreadable)**	**F1 Score (Spreadable)**	**Accuracy (Spreadable)**
Xception	0.666	69.7248	0.916	91.4634
VGG_19	0.608	62.8440	0.86	85.9756
ResNet152	**0.826**	**85.3211**	**1**	**100.0000**
InceptionV3	0.578	62.8440	0.89	89.0244
MobileNet	0.704	73.3945	0.976	97.5610
Densenet169	0.816	83.0275	0.983	98.1707
NasNetMobile	0.32	44.4954	0746	75.6098
EfficientNetB0	0.894	90.3670	0.993	99.3902
EfficientNetB1	0.896	90.8257	0.993	99.3902
EfficientNetV2B2	**0.86**	**87.6147**	**1**	**100.0000**
ConvNeXtXlarge	**0.904**	**92.2018**	**1**	**100.0000**

TABLE IV. F1 SCORE AND ACCURACIES OF LETTUCE DATASETS

Models	**F1 Score(Spreadable)**	**Accuracy (Spreadable)**	**F1 Score(Non Spreadable)**	**Accuracy(Non-Spreable)**
Xception	0.751	77.0889	1.00	100.0000
VGG_19	0.744	76.8194	0.993	99.2366
ResNet152	0.945	95.1482	1.00	100.0000
InceptionV3	0.764	77.6280	0.926	92.3664
MobileNet	0.677	72.2371	0.993	99.2366
Densenet169	0.955	96.2264	1.00	100.0000
NasNetMobile	0.710	71.6981	0.96	96.1832
EfficientNetB0	0.820	94.0700	0.97	96.9465
EfficientNetB1	0.935	94.0700	0.993	99.2366
EfficientNetV2B2	0.942	94.6091	1.00	100.0000
ConvNeXtXLarge	**0.961**	**96.4959**	**1.00**	**100.0000**

TABLE V. ACCURACY OF BANANA DATASET OF ALL KINDS OF PARTS OF THE PLANT

Models	Accuracy (Stem)	Accuracy (node)	Accuracy (Leaf)	Accuracy (Fruit)	Accuracy (Non-Infectious)
Xception	87.4477	100.000	85.6557	60.344	93.3824
VGG_19	80.3347	100.000	82.7869	93.103	81.6176
ResNet152	94.1423	100.000	98.3607	100.00	99.2647
InceptionV3	82.0084	100.000	82.3770	98.275	94.1176
MobileNet	88.2845	100.000	94.6721	51.724	99.2647
Densenet169	93.3054	100.000	98.3607	100.0000	99.2647
NasNetMobile	**44.3515**	**92.2078**	**74.1803**	**94.827**	**76.4706**
EfficientNetB0	94.1423	100.000	95.4918	100.0000	98.5294
EfficientNetB1	94.9791	100.000	97.1311	100.0000	98.5294
EfficientNetV2B2	95.3975	100.000	98.3607	100.0000	99.2647
ConvNeXtXLarge	**95.3975**	**99.3506**	**99.5902**	**98.275**	**100.0000**

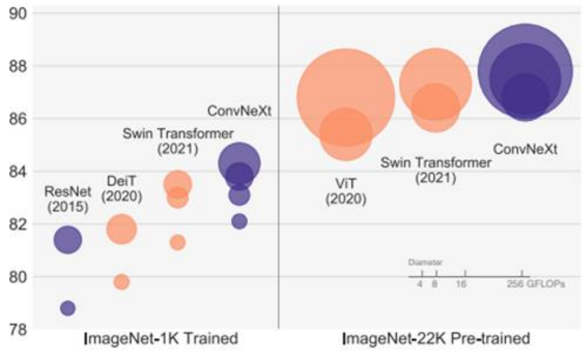


Fig. 15. Classification results for ConvNets and vision Transformers [38].

The accuracy achieved after combining all disease categories from the datasets of lettuce, soybean, and banana, which include both infectious and non-infectious conditions, is displayed in Tables VI through Table VIII.

EfficientNetV2B2 has the maximum accuracy of 95.81% from the Table VI, compared to a baseline accuracy of 56.38% from the NASNetMobile model.

C. *What is EfficientNetV2?*

The successor to EfficientNets is EfficientNetV2. EfficientNet is a family of models that was unveiled in 2019 and is optimised for FLOPs and parameter efficiency [39]. It makes use of neural architecture search to find the EfficientNet-B0 baseline model with the best possible accuracy and FLOPs trade-off. EfficientNets aggressively increase picture size, which results in high memory use and slow training. To overcome this problem, slightly altered the scaling rule and limited the maximum picture size to a lower amount. EfficientNetV2's technology, Deep learning models and training set both are becoming bigger and bigger. Efficiency in training is crucial in this situation. For instance, few-shot learning is demonstrated by the GPT-3 model, which has unheard-of model and training data volumes. However, retraining or enhancing the model is challenging because it takes weeks to train with thousands of GPUs. To construct this model, it combines scaling and training-aware Neural Architecture Search (NAS)to maximize training time and parameter effectiveness.

The findings of the models are remarkably comparable to those of the lettuce dataset, as can be seen from the Table VII, where EfficientNetV2B2 achieves a 93.97% accuracy while NASNetMobile achieves a baseline accuracy of 40.66%.

TABLE VI. ACCURACY OF THE COMBINED DATASET(SPREADABLE + NON SPREADABLE) OF LETTUCE

Models	F1 Score	Accuracy(Lettuce)
Xception	**0.7566**	**77.3128**
VGG_19	**0.7366**	**73.3480**
ResNet152	0.9411	94.7137
InceptionV3	0.7794	79.7357
MobileNet	0.8477	85.4626
Densenet169	0.9522	95.3744
NasNetMobile	**0.5533**	**56.3877**
EfficientNetB0	0.9511	95.3744
EfficientNetB1	0.9544	95.5947
EfficientNetV2B2	**0.9555**	**95.8150**
ConvNeXtXLarge	**0.9511**	**95.1542**

TABLE VII. ACCURACY OF THE COMBINED DATASET(SPREADABLE + NON SPREADABLE) OF SOYBEAN

Models	F1 Score	Accuracy(Soybean)
Xception	**0.7028**	**72.5904**
VGG_19	**0.6585**	**69.5783**
ResNet152	0.9142	93.3735
InceptionV3	0.6914	70.4819
MobileNet	0.6971	72.5904
Densenet169	0.8600	88.2530
NasNetMobile	**0.3414**	**40.6627**
EfficientNetB0	0.9057	92.1687
EfficientNetB1	0.8828	90.3614
EfficientNetV2B2	**0.9242**	**93.9759**
ConvNeXtXLarge	**0.9185**	**93.3735**

TABLE VIII. ACCURACY OF THE COMBINED DATASET (SPREADABLE + NON SPREADABLE) OF BANANA

Models	F1 Score	Accuracy(Banana)
Xception	**0.9362**	**95.1865**
VGG_19	**0.8877**	**89.7714**
ResNet152	0.9762	98.3153
InceptionV3	0.9408	95.9085
MobileNet	0.9677	97.1119
Densenet169	0.9746	98.0746
NasNetMobile	**0.6562**	**65.3430**
EfficientNetB0	0.9608	97.2323
EfficientNetB1	0.9685	97.7136
EfficientNetV2B2	**0.8992**	**98.0746**
ConvNeXtXLarge	**0.9100**	**95.3069**

The Table VIII makes it very evident that ResNet model has the maximum accuracy of 98.31% whereas NASNetMobile Baselines are at 65.34%.

D. What is ResNet?

ResNet was created with the goal of resolving computer vision issues. Deep residual networks that take advantage of remaining blocks to increase model precision. The concept of "skip connections," which is the foundation of the residual blocks, is the strength of this type of neural network.

1) 'Skip Connections' in ResNet: There are two ways that these skip connections work. The gradient is given a new shortcut to employ in order to address the issue of the fading gradient. Additionally, they give the model the capacity to pick up an identity function. This ensures that the performance of the model's top tiers is equal to or better than that of its lower layers. In conclusion, the residual blocks lets the layers acquire identity functions considerably more quickly. ResNet therefore decreases errors while boosting the efficiency for deep neural networks with far more neural layers. In other words, the skip connections integrate the outputs of older layers with outputs from stacked layers, enabling the training of networks that are far deeper than was previously possible.

Final point: ResNet, sometimes referred to as residual network, was a crucial development that changed how deep convolutional neural networks are trained for computer vision tasks. The venerable Resnet featured 34 layers with 2-layer bottleneck blocks, while more advanced models, like the Resnet50, used 3-layer bottleneck blocks that guarantee greater accuracy and faster training.

Tables VI to VIII shows the models in bold that are being examined for improvement of outcomes by modifying a hyperparameter, particularly the optimizer.

The optimizers that are considered for research work are:

- Adadelta
- Adagrad
- Adam
- Adamax
- Ftrl
- Nadam
- RMSprop
- SGD

2) Stochastic Gradient Descent (SGD): This is a typical 'base' optimizer, and many others are variations on it [40],[41]. It is adjustable by varying the learning rate, momentum and decay.

a) Learning rate: The learning rate controls the magnitude of parameter updates at each iteration of the optimization algorithm. A higher learning rate allows for larger updates, potentially leading to faster convergence but also increasing the risk of overshooting the optimal solution. On the other hand, a lower learning rate results in smaller updates, which may slow down convergence but can help the model settle into a more accurate and stable solution.

b) Momentum: propels SGD in the desired direction while dampening oscillations. Essentially, it allows SGD to push past local optima, resulting in quicker convergence and reduced oscillation. A normal momentum value is between 0.5 and 0.9.

c) Decay: For the learning rate, you can provided a decay function. As training advances, this will alter the learning rate. Decay functions are- Time delay, Step delay and Exponential delay.

d) Nesterov: Nesterov momentum is a variant of the momentum method that provided better theoretical converge guarantees for convex functions. In practice, it is somewhat more effective than conventional momentum.

3) Adaptive learning rate optimizers

a) Adagrad: Adagrad is an optimizer with variable parameter-specific learning rates based on how frequently a parameter is altered during training [42].

b) Adadelta: Adadelta is an optimizer that dynamically adapts the learning rate during training without the need for a predefined initial learning rate. It uses a combination of the gradient information and a moving average of the past gradients to adjust the learning rate at each iteration, allowing for efficient convergence [43]. The learning rate in Adadelta is not explicitly set by the user but is internally calculated based on the algorithm's parameters and the gradient history.

c) RMSprop: RMSprop, like Adadelta, modifies the Adagrad technique in a very easy way to lessen its aggressive, monotonically declining learning rate [44].

d) Adam: Adam is an RMSProp optimizer update. It's essentially RMSprop with momentum [45].

e) Adamax: It is a first-order gradient-based optimization approach and a version of Adam based on the infinite norm. It is well suited to learning time-variant processes, such as voice data with dynamically changing noise circumstances, because to its capacity to alter the learning rate based on data features.

f) Nadam: Similarly, to how Adam is RMSprop with momentum, Nadam is Adam with Nesterov momentum.

4) Ftrl: "Follow The Regularized Leader" (FTRL) is an optimization technique created by Google in the early 2010s for click-through rate prediction [46]. It works well with shallow models with vast and sparse feature areas, this was discussed by McMahan et al. (2013). Both online L2 regularization (the L2 regularization described in the study above) and shrinkage-type L2 regularization are supported in the Keras version (which is the addition of an L2 penalty to the loss function).

The results of the cluster of models are not as linear as shown in Tables IX to XI, and the behavior of each model with its corresponding optimizer differs for different datasets.

The adamax optimizer dominates in every other model, whereas SGD optimizers have low efficiency rate due to its poor processing performance. SGD is a very fundamental technique that is seldom employed in applications nowadays. Another issue with the method is, its constant learning rate for each epoch. Furthermore, it is not particularly good at dealing with saddle points. Because of the frequent modifications in the learning rate, Adagrad performs better than stochastic gradient descent in general. It works well when dealing with sparse data. RMSProp produces comparable results to the gradient descent technique using momentum; the only difference is how the gradients are calculated. Finally, the Adam optimizer inherits the best aspects of RMSProp and other algorithms.

The Adamax optimizer provideds a faster computation time, provided better results than other optimization techniques, it requires fewer tuning parameters. Adam is recommended as default optimizer for majority of applications as a consequence of all of this. Any application may have the highest chance of producing the finest outcomes if Adamax optimizer is used.

Finally, we discovered that even Adamax optimizer had certain drawbacks. In some circumstances, algorithms like as SGD may be more useful and perform better than the Adam optimizer. To pick the finest optimization method and obtain great results, it is critical to understand the needs and the type of data dealt with.

Since the performance of NASNet model with Adamax as optimizer , VGG19 model with Adagrad as optimizers and Xeception model with Adamax as optimizer for Lettuce dataset NASNet model with Ftrl as optimizer, VGG19 model with Adagrad as optimizer and Xeception model with Adamax as optimizer for soybean dataset and NASNet model with Adam as optimizer, Xeception model with Adamax as optimizer for Banana dataset, although is highest with respect to other optimizer, yet its accuracies are incompatible for practical consideration.

Hence it is required to improve its performance.

One way to enhance performance is to train the model so that it is exclusively prepared for this dataset by initializing the weights to 0.

TABLE IX. ACCURACY OF MODELS FOR DIFFERENT OPTIMIZERS (LETTUCE DATASET)

Lettuce Dataset			
Sl No	**Model**	**Optimizer**	**Accuracy**
1	ConvNeXtXtLarge	Adadelta	80.17621
2	ConvNeXtXtLarge	Adagrad	92.73128
3	ConvNeXtXtLarge	Adam	95.15419
4	**ConvNeXtXtLarge**	**Adamax**	**96.9163**
5	ConvNeXtXtLarge	Ftrl	20.26432
6	ConvNeXtXtLarge	Nadam	92.95154
7	ConvNeXtXtLarge	RMSprop	95.15419
8	ConvNeXtXtLarge	SGD	92.73128
9	EfficientNetV2B2	Adadelta	32.59912
10	EfficientNetV2B2	Adagrad	89.86784
11	EfficientNetV2B2	Adam	95.81498
12	EfficientNetV2B2	Adamax	95.81498
13	EfficientNetV2B2	Ftrl	95.81498
14	EfficientNetV2B2	Nadam	95.81498
15	**EfficientNetV2B2**	**RMSprop**	**96.25551**
16	EfficientNetV2B2	SGD	89.86784
17	NASNetMobile	Adadelta	10.79295
18	NASNetMobile	Adagrad	36.78414
19	NASNetMobile	Adam	56.38767
20	**NASNetMobile**	**Adamax**	**58.81057**
21	NASNetMobile	Ftrl	56.38767
22	NASNetMobile	Nadam	56.38767
23	NASNetMobile	RMSprop	41.18943
24	NASNetMobile	SGD	46.69604
25	VGG_19	Adadelta	64.97797
26	**VGG_19**	**Adagrad**	**84.14097**
27	VGG_19	Adam	74.6696
28	VGG_19	Adamax	83.70044
29	VGG_19	Ftrl	20.26432
30	VGG_19	Nadam	78.19383
31	VGG_19	RMSprop	22.68722
32	VGG_19	SGD	9.69163
33	Xception	Adadelta	30.17621
34	Xception	Adagrad	68.06167
35	Xception	Adam	75.11013
36	**Xception**	**Adamax**	**83.48018**
37	Xception	Ftrl	20.26432
38	Xception	Nadam	73.78855
39	Xception	RMSprop	78.85463
40	Xception	SGD	35.46256

TABLE X. ACCURACY OF MODELS FOR DIFFERENT OPTIMIZERS (SOYBEAN DATASET)

Soybean Dataset			
Sl No	**Model**	**Optimizer**	**Accuracy**
1	ConvNeXtXtLarge	Adadelta	82.22892
2	ConvNeXtXtLarge	Adagrad	89.15663
3	ConvNeXtXtLarge	Adam	93.37349
4	**ConvNeXtXtLarge**	**Adamax**	**94.57831**
5	ConvNeXtXtLarge	Ftrl	14.71698
6	ConvNeXtXtLarge	Nadam	93.1677
7	ConvNeXtXtLarge	RMSprop	93.6747
8	ConvNeXtXtLarge	SGD	90.66265
9	EfficientNetV2B2	Adadelta	93.9759
10	EfficientNetV2B2	Adagrad	88.55422
11	EfficientNetV2B2	Adam	93.9759
12	EfficientNetV2B2	Adamax	92.46988
13	EfficientNetV2B2	Ftrl	86.74699
14	**EfficientNetV2B2**	**Nadam**	**96.08434**
15	EfficientNetV2B2	RMSprop	94.57831
16	EfficientNetV2B2	SGD	88.55422
17	NASNetMobile	Adadelta	12.3494
18	NASNetMobile	Adagrad	31.3253
19	NASNetMobile	Adam	38.55422
20	NASNetMobile	Adamax	43.07229
21	**NASNetMobile**	**Ftrl**	**43.9759**
22	NASNetMobile	Nadam	43.6747
23	NASNetMobile	RMSprop	40.66265
24	NASNetMobile	SGD	40.66265
25	VGG_19	Adadelta	59.33735
26	**VGG_19**	**Adagrad**	**80.72289**
27	VGG_19	Adam	58.43373
28	VGG_19	Adamax	70.18072
29	VGG_19	Ftrl	23.79518
30	VGG_19	Nadam	68.9759
31	VGG_19	RMSprop	24.6988
32	VGG_19	SGD	16.26506
33	Xception	Adadelta	40.66265
34	Xception	Adagrad	64.75904
35	Xception	Adam	66.86747
36	**Xception**	**Adamax**	**74.6988**
37	Xception	Ftrl	19.27711
38	Xception	Nadam	73.19277
39	Xception	RMSprop	70.48193
40	Xception	SGD	48.79518

TABLE XI. ACCURACY OF MODELS FOR DIFFERENT OPTIMIZERS (BANANA DATASET)

Banana Dataset			
Sl No	**Model**	**Optimizer**	**Accuracy**
1	ConvNeXtXtLarge	Adadelta	95.29653828
2	**ConvNeXtXtLarge**	**Adagrad**	**98.31528279**
3	ConvNeXtXtLarge	Adam	95.4356249
4	ConvNeXtXtLarge	Adamax	97.35258724
5	ConvNeXtXtLarge	Ftrl	95.30685921
6	ConvNeXtXtLarge	Nadam	95.16727657
7	ConvNeXtXtLarge	RMSprop	95.3078993
8	ConvNeXtXtLarge	SGD	95.30685921
9	EfficientNetV2B2	Adadelta	43.92298436
10	EfficientNetV2B2	Adagrad	97.83393502
11	EfficientNetV2B2	Adam	97.71359807
12	**EfficientNetV2B2**	**Adamax**	**98.0746089**
13	EfficientNetV2B2	Ftrl	85.19855596
14	EfficientNetV2B2	Nadam	97.95427196
15	EfficientNetV2B2	RMSprop	97.87426738
16	EfficientNetV2B2	SGD	97.95427196
17	NASNetMobile	Adadelta	8.664259928
18	NASNetMobile	Adagrad	37.18411552
19	**NASNetMobile**	**Adam**	**88.56799037**
20	NASNetMobile	Adamax	15.04211793
21	NASNetMobile	Ftrl	13.35740072
22	NASNetMobile	Nadam	86.64259928
23	NASNetMobile	RMSprop	69.7954272
24	NASNetMobile	SGD	36.70276775
25	**VGG_19**	**Adadelta**	**98.31528279**
26	VGG_19	Adagrad	98.0746089
27	VGG_19	Adam	77.61732852
28	VGG_19	Adamax	67.99037304
29	VGG_19	Ftrl	14.92178099
30	VGG_19	Nadam	90.49338147
31	VGG_19	RMSprop	76.89530686
32	VGG_19	SGD	8.784596871
33	Xception	Adadelta	31.28760529
34	Xception	Adagrad	47.17208183
35	Xception	Adam	90.97472924
36	**Xception**	**Adamax**	**93.50180505**
37	Xception	Ftrl	13.47773767
38	Xception	Nadam	87.36462094
39	Xception	RMSprop	89.89169675
40	Xception	SGD	39.95186522

Why didn't the process initially explore retraining or fine-tuning the model from scratch?

Due to the small amount of data, creating new models from start would be a resource and time-intensive operation with no assurance of performance. These models were previously trained quite effectively. In order to improve performance efficacy in our case, it is thus preferable practice to load those pertained models and use the information that the two models have previously acquired in the course of their original work. Transfer learning and fine-tuning are commonly confused with one another since they are both parts of the same process. Many refer to the entire process as fine-tuning since we often do so after transfer learning.

However, fine-tuning involves more than just applied the weights from the pre-trained models. In order to adjust the model to the present job, it is also using prior information but freezing some layers while training the last layers at a slow learning rate. Convolution deep learning model results are shown to provide a better understanding of the entire process.

TABLE XII. ACCURACIES OF MODELS BEFORE AND AFTER RETRAINING WITH THE DATASETS

Sl No	Model	Optimizer	Accuracy before retraining	Accuracy after retraining
		Lettuce Dataset		
1	NASNetMobile	Adamax	58.81057269	71.36564
2	VGG_19	Adagrad	84.14096916	90.30837
3	**Xception**	**Adamax**	**83.48017621**	**96.69604**
		Soybean Dataset		
4	NASNetMobile	Ftrl	43.97590361	19.27711
5	VGG_19	Adagrad	80.72289157	82.22892
6	**Xception**	**Adamax**	**74.69879518**	**92.46988**
		Banana Dataset		
7	NASNetMobile	Adam	88.56799037	33.81468
8	**Xception**	**Adamax**	**93.50180505**	**98.19495**

E. Concept of Underfitting and Overfitting

Why poor accuracy is viewed for few models over other models with various optimizers?

If a model adequately generalizes all new input data from the problem domain, it is considered to be a good machine learning model. Additionally, underfitting and overfitting are the main reasons why machine learning algorithms perform poorly [47].

F. Concept of Bias, Variance, Underfitting and Overfitting

1) Bias: Essentially, it is the error rate of the training data. Whenever the error margin is high, we say the bias is strong, while when it is low, the bias is low.

2) Variance: The variance is the difference in the error margin between the training and test sets of data. The variance is described as being high when it is large and low whenever the difference between both the errors is small. Usually, we want to expand our model with the least amount of variance.

3) Underfitting: Underfitting is the term used whenever a statistical model as well as machine learning algorithm fails to capture the overall pattern of the data, i.e., when it performs well on training data but poorly on testing data. Its recurrence simply shows that model or method doesn't really adequately fit the data. It frequently happens when there are not enough data to build a solid model or when we try to build a linear model with too little non-linear data. Because its rules are too basic and flexible to be applied to such scant data, a machine learning model will probably make a number of inaccurate predictions under these circumstances. Underfitting may be avoided by utilizing more data and restricting the features through feature selection [48].

Underfitting, is when a model is unable to perform satisfactorily on the training dataset or generalize to new data.

G. Justifications for underfitting

- Low variance and high covariance.
- The used training dataset's size is insufficient.
- The model is rather basic.
- Training data has noise in it and is not being eliminated.

Methods to reduce underfitting

- Amplify model complexity.
- Boost feature count by doing feature engineering.
- Data noise should be removed.
- To achieve better outcomes, increase the period of training or the number of epochs.

1) Overfitting: An overfitted statistical model is one that cannot accurately predict events, based on test data [49]. When a model has been trained with a massive quantity of data, it begins to gain knowledge from the disturbance and incorrect data entries in the given dataset and when test data is used for testing, there is a lot of diversity. The model is unable to correctly recognize the data because of the overabundance of characteristics and distortion. Since these give machine learning algorithms greater freedom to build the model depending on the dataset, non-parametric and non-linear techniques are the primary sources of overfitting and can result in extremely illogical models [50]. Using a linear approach to analyze linear data is one strategy to avoid overfitting.

Overfitting, is a problem when the evaluation of machine learning algorithms on unknown data varies from the analysis on training data.

Overfitting has the following causes:

- Both variance and bias are high.
- The model is very sophisticated.
- The volume of training data.

Methods to reduce overfitting

- Expand the training data.
- Simplify the model.
- During the training phase usage of early stopping stay updated on the loss during the training period, and cease training as soon as it starts to rise.
- Regularization of the Ridge and the Lasso [51].
- To mitigate overfitting in neural networks, using a dropout technique.

H. How to Fit a Statistical Model Well?

When a statistical model makes predictions that are error-free, that is the ideal situation, it is said to be a good match with the data. This situation may exist anywhere between overfitting and underfitting. To understand the model, we need to look at how it performs over time as it learns from the training dataset.

As time goes on, the model would continue to learn, as a result, the model's accuracy just on test & training data will decrease over time. If the model is given an abnormally lengthy time to learn, the accumulation of junk and less important characteristics can make it more susceptible to overfitting. The model's performance will therefore suffer. In order to obtain a good match, you must stop just as the error starts to grow worse. In both our concealed testing dataset & training datasets, the model is judged as being competent at this point.

I. How to find whether the model chosen is overfit or underfit?

When the validation accuracy increases after retraining and subsequently sharply decreases, the model is overfit. While in the event of underfit, there is just a slow, lower value rise in validation accuracy.

Models with excellent accuracy prior to retraining but significantly reduced accuracy after retraining are shown in Table XII. The validation accuracy and test accuracy in Table XIII are used to determine whether the model is overfit or underfit.

The model training details show that none of the models under consideration are overfit, but NASNetMobile model with Ftrl as optimizer for soybean dataset and NASNetMobile model with Adam as optimizer for banana dataset is underfit. This is because the mentioned model has good training accuracy but low validation and test accuracy making it underfit.

Also, the presence of overfit model does not affect the accuracies in present research case because of consideration of best fit model and early stopping criteria.

TABLE XIII. ACCURACIES OF MODEL WITH VALIDATION ACCURACIES

Sl.No	Model	Optimizer	Accuracy (when retrained)	Validation accuracy (when retraing)	Accuracy (after retraining)
Lettuce Dataset					
1	NASNet Mobile	Adamax	99.49	73.951	71.365
2	VGG_19	Adagrad	99.93	87.23	90.308
3	Xception	Adamax	100	96.689	96.696
Soybean Dataset					
4	**NASNet Mobile**	**Ftrl**	**75.89**	**19.219**	**19.277**
5	VGG_19	Adagrad	100	83.784	82.228
6	Xception	Adamax	99.61	96.096	92.469
Banana Dataset					
7	**NASNet Mobile**	**Adam**	**92.9**	**33.454**	**33.814**
8	Xception	Adamax	96.75	91.095	98.194

VI. Proposed Methodology

In the context of proposing a methodology for developing a high-accuracy convolutional neural network (CNN) model, particularly for tasks like image classification, the method integrates cutting-edge architectural principles from EfficientNetV2B2 with custom-designed elements, specifically a novel activation function and optimizer. Here's a detailed breakdown of the proposed methodology.

To further elucidate the rationale behind choosing the specific combination of ReLU and Tanh for the custom activation function and the integration of Adam and SGD characteristics in the custom optimizer, we can refer to the previously discussed implementation and general principles in neural network optimization and activation functions as shown in Fig. 16.

A. Components of the Methodology

1) Base Architecture (EfficientNetV2):

- EfficientNetV2B2 [52] is chosen as the foundational architecture due to its state-of-the-art performance in image classification tasks.
- It utilizes a compound scaling method that uniformly scales the depth, width, and resolution of the network, making it highly efficient and effective.

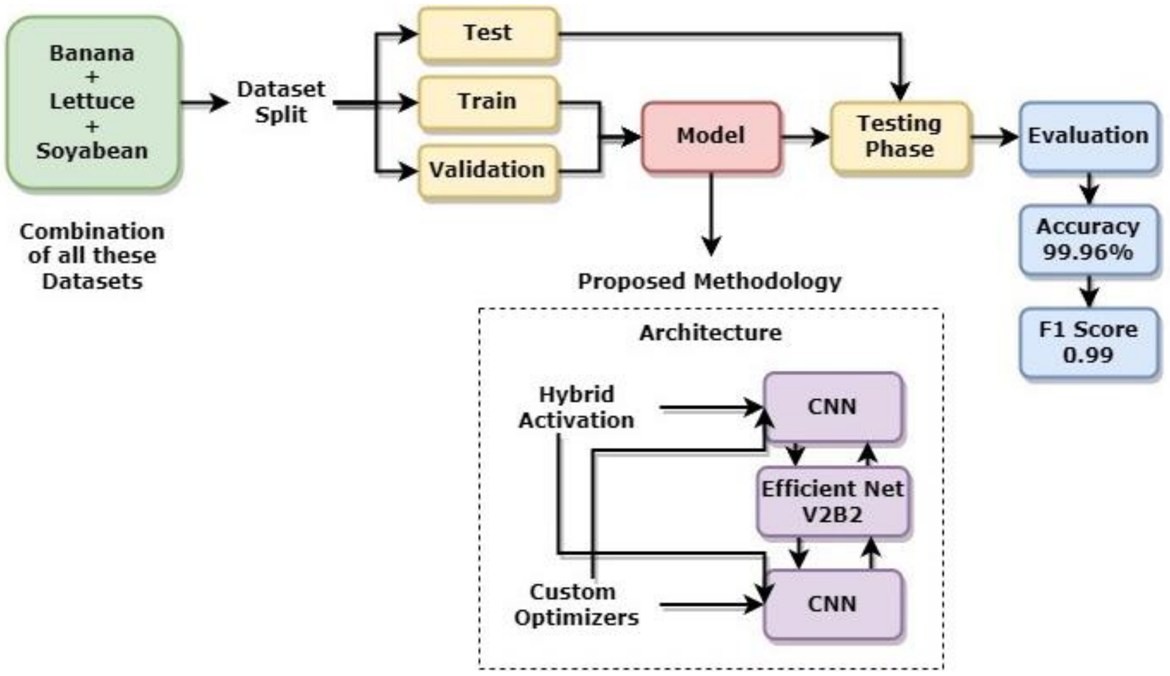


Fig. 16. Proposed methodology.

2) Custom Activation Function:

- A novel activation function is introduced to potentially improve the learning process.
- The function, custom activation, is a combination of ReLU and Tanh functions. ReLU ensures non-saturation of gradients for positive values, enhancing learning speed, while Tanh provided output normalization, potentially aiding in stabilizing the learning process.

3) Custom Optimizer:

- Developing a custom optimizer aims to enhance the training efficiency and convergence rate.
- The optimizer combines elements of Adam (adaptive learning rates) and SGD (stochastic gradient descent), attempting to utilize the benefits of both.

4) Integration and Training:

- The EfficientNetV2B2 base model is loaded with pre-trained ImageNet weights to leverage transfer learning, accelerating the training process and improving initial accuracy.
- The top layers of the model are replaced with custom layers, including Global Average Pooling and Dense layers, utilizing the custom activation function.
- The model is compiled with the custom optimizer, and categorical cross-entropy is used as the loss function, suitable for multi-class classification tasks.

5) Training Strategy:

- Initially, the EfficientNetV2B2 base layers are frozen to preserve the pre-trained features, and only the custom top layers are trained.
- Subsequently, fine-tuning can be performed by unfreezing some of the top layers of the base model and continuing the training, allowing for refined feature extraction tailored to the specific dataset.

6) Evaluation:

- The model's performance is evaluated using accuracy metrics on a validation dataset.
- Regular checkpoints and monitoring are employed to track the training progress and prevent overfitting.

B. Integration of Concepts Based on Previous Results

The decision to combine these specific elements from ReLU, Tanh, Adam, and SGD is not only based on their individual strengths but also on empirical observations from previous implementations:

1) ReLU and Tanh: The combined use of ReLU and Tanh in various architectures has shown promising results in terms of faster convergence and improved accuracy, as these functions complement each other's properties.

2) Adam and SGD: Similarly, the integration of Adam's adaptive learning rate and SGD's generalization capabilities aims to create a more robust and efficient optimizer. This is based on observations where models trained with Adam initially show rapid improvement but sometimes fail to achieve the level of generalization that SGD can provided.

3) Outcomes:

- Enhanced Model Performance: By integrating the architectural efficiency of EfficientNetV2 with the novel elements of the custom activation function and optimizer, the model has demonstrated remarkable performance in image classification tasks. Notably, this approach has achieved a remarkable accuracy of 99.96%, positioning it at the forefront of current image classification models. This high level of accuracy indicates an exceptional ability of the model to correctly classify images, minimizing both false positives and false negatives.

- Superior F1 Score: Alongside accuracy, the model has achieved an F1 score of 0.99. The F1 score is a more complex metric that considers both precision and recall, providing a more holistic view of the model's performance. An F1 score of 0.99 implies that the model not only accurately classifies the positive cases but also maintains a high rate of successfully identifying true negatives. This balance is crucial in scenarios where both types of classification errors carry significant consequences.
- Improved Learning Dynamics: The custom activation function and optimizer have played a pivotal role in enhancing the training process. These custom elements have contributed to improved training stability and accelerated convergence speed, enabling the model to quickly adapt and optimize its performance. The combination has been instrumental in achieving the high accuracy and F1 score, underlining the effectiveness of these custom components in handling complex learning tasks.

VII. Conclusion

This paper focuses on the use of transfer learning for the identification of spreadable and non-spreadable plant diseases. The study considers three different plant types, namely lettuce, soybean, and banana, and addresses the classification of the most prevalent diseases in these plants. The diseases are categorized into spreadable and non-spreadable diseases, treated as distinct classes in the analysis.

To evaluate the classification accuracy of different models, a comparative research approach is employed. The performance of 11 transfer learning models available in Keras are assessed on separate datasets for spreadable and non-spreadable diseases. Additionally, the models are evaluated on a combined dataset that includes five different portions of the plants, comprising both healthy and diseased parts. Early stopping criteria are set at a minimum of 20 to 30 epochs with a patience of 6 for comparison. It is observed that the metrics and accuracy of the models vary depending on the dataset being used. However, some of the selected models did not exhibit the anticipated high accuracy after training on the datasets.

To improve the model performance, various techniques such as optimizing the models and retraining them from scratch can be employed. These strategies aim to enhance the accuracy and effectiveness of the classification models for identifying spreadable and non-spreadable plant diseases. The available optimizers in Keras are taken into consideration in order to increase accuracy, that includes SGD, RMSprop, Adam, Adadelta, Adagrad, Adamax, Nadam, and Ftrl. However, this strategy only worked for higher models (like EfficientNet models, ConvNeXt); smaller models (like VGG-19, Xception) showed less improvement. The paper's major goal was to select a lower model since it is less complicated to train for and has a smaller width. Improving them suggests an upgrade to the foundational CNN model, making the study more flexible. With certain models, there is a rapid fall in accuracy, leading it to be considered as either underfit or overfit. In the current instance, the NASNetMobile model is underfit, and situations of overfit are not evident because of the early stopping approach. VGG_19 model with Adadelta as optimizer without retraining and Xception model with Adamax as optimizer when retrained from scratch, outperform in terms of classification metrics for the datasets under consideration.

In addition to these strategies, the paper proposed a novel methodology focusing on the integration of an EfficientNetV2-style architecture with a custom-designed activation function and optimizer. The custom activation function, a hybrid of ReLU and Tanh, aims to enhance learning dynamics by combining the benefits of non-saturation (from ReLU) and output normalization (from Tanh). The custom optimizer, blending elements of Adam and SGD, is designed to achieve a balance between adaptive learning rates and effective generalization. This proposed methodology, especially with the EfficientNetV2's efficient scaling and advanced architecture, is expected to yield even higher accuracy and robustness in classifying plant diseases. Notably, this approach has achieved remarkable performance with an accuracy of 99.96% and an F1 score of 0.99 in the classification tasks, setting a new standard in the field and underscoring the effectiveness of combining advanced neural network architectures with innovative custom components for complex classification challenges.

VIII. Future Work

Moving forward, several key areas offer promising opportunities to extend and enhance the research presented in this study. One significant direction is the expansion and diversification of the dataset. By including a broader range of plant species, diseases, and environmental conditions, the models could be made more robust and generalizable across different agricultural contexts. Additionally, incorporating data from various geographical regions and employing data augmentation techniques could help address issues of overfitting and improve the model's performance on smaller or imbalanced datasets.

Integrating real-time data from environmental sensors is another avenue that could significantly enhance the predictive accuracy of the models, especially in relation to both biotic and abiotic plant stressors. By developing models capable of adapting to dynamic environmental conditions, the relevance and effectiveness of AI-driven solutions in agriculture could be substantially improved. Moreover, refining the custom activation function and optimizer introduced in this study remains an important task. Testing these components across different deep learning architectures and applications, such as pest detection or crop yield prediction, could assess their versatility and broader applicability.

Ethical considerations and societal impacts also warrant close attention. As AI-driven plant disease identification systems are deployed, it is crucial to address potential ethical issues, such as data privacy, fairness, and the implications for small-scale farmers. Moreover, the societal impacts, including potential job displacement and the need for upskilling agricultural workers, should be carefully considered to ensure responsible and equitable deployment of these technologies.

Finally, exploring the scalability of the proposed models for large-scale farming operations is essential, particularly in terms of computational efficiency and resource constraints. Investigating cloud-based or edge-computing solutions could facilitate real-time disease detection in remote or resource-limited settings. Collaborative, multi-disciplinary research involving AI experts, agronomists, plant pathologists, and agricultural economists will be critical in developing holistic solutions that effectively address the complexities of plant disease management.

CONFLICTS OF INTEREST

The authors declare no conflict of interest.

FUNDING STATEMENT

This research did not receive any specific funding or financial support.

ETHICAL STATEMENT

This research was conducted in accordance with ethical principles, including informed consent, confidentiality, and adherence to relevant guidelines.

AUTHORS' CONTRIBUTION

Conceptualization: Asha Rani K P and Gowrishankar S jointly conceived and designed the research project.

Methodology: Asha Rani K P developed the methodology for the study.

Validation: Asha Rani K P and Gowrishankar S collectively validated the results and ensured their accuracy.

Formal Analysis: Asha Rani K P conducted the formal analysis of the data.

Investigation: Asha Rani K P and Gowrishankar S carried out the investigation and collected the necessary data.

Resources: Asha Rani K P and Gowrishankar S provided the required resources for the project.

Data Curation: Asha Rani K P curated and prepared the dataset for analysis.

Writing—Original Draft Preparation: Asha Rani K P wrote the initial draft of the manuscript.

Writing—Review and Editing: Gowrishankar S critically reviewed and Asha Rani K P edited the manuscript.

Visualization: Asha Rani K P created the visualizations used in the paper.

Both the authors have substantially contributed to the work reported and have approved the final version of the manuscript.

DATA AVAILABILITY STATEMENT

Images used for the study are obtained from CrowdAI [27] and PlantVillage dataset [28].

REFERENCES

[1] Raid, Richard N. "Lettuce Diseases and Their Management." Diseases of Fruits and Vegetables: Volume II, n.d., 121–47.

[2] Carrasco, Gilda A., and S. W. Burrage. "Diurnal Fluctuations in Nitrate Accumulation and Reductase Activity In Lettuce (LACTUCA SATIVA L.) Grown using Nutrient Film Technique" Acta Horticulturae, no. 323, International Society for Horticultural Science (ISHS), Feb. 1993, pp. 51–60. Crossref, https://doi.org/10.17660/actahortic.1993.323.3.

[3] Singh, Gaurav, Garima Dukariya, and Anil Kumar. "Distribution, Importance and Diseases of Soybean and Common Bean: A Review." Biotechnology Journal International, 2020, 86–98.

[4] Wahome, C. N., Maingi, J. M., Ombori, O., Kimiti, J. M., &amp; Njeru, E. M. (2021). Banana production trends, cultivar diversity, and tissue culture technologies uptake in Kenya. International Journal of Agronomy, 2021, 1–11. https://doi.org/10.1155/2021/6634046

[5] K . Lakshmi Narayanan ,1 R. Santhana Krishnan ,2 Y. Harold Robinson , E. Golden Julie , S. Vimal , V. Saravanan , and M. Kaliappan5. "Banana Plant Disease Classification Using Hybrid Convolutional Neural Network"

[6] Andreas Kamilaris, Francesc X. Prenafeta-Boldú, "Deep learning in agriculture: A survey", Computers and Electronics in Agriculture, Volume 147, Pages 70-90, ISSN 0168-1699, 2018.

[7] Ramcharan, Amanda, Kelsee Baranowski, Peter McCloskey, Babuali Ahmed, James Legg, and David P. Hughes. "Deep Learning for Image-Based Cassava Disease Detection." Frontiers in Plant Science 8 (2017)

[8] N.Saranya, L. Pavithra, N. Kanthimathi, B. Ragavi, P. Sandhiyadevi . "Detection of Banana Leaf and Fruit Diseases Using Neural Networks"

[9] Michael Gomez Selvaraj, Alejandro Vergara, Frank Montenegro , Henry Alonso Ruiz, Nancy Safari , Dries Raymaekers , Walter Ocimati , Jules Ntamwira , Laurent Tits , Aman Bonaventure Omondi , Guy Blomme "Detection of banana plants and their major diseases through aerial images and machine learning methods: A case study in DR Congo and Republic of Benin".

[10] E. Miao, Guixia Zhou, and Shengxue Zhao "Research on Soybean Disease Identification Method Based on Deep Learning"

[11] Sachin B. Jadhav, Vishwanath R. Udupi, Sanjay B. Patil, "Soybean leaf disease detection and severity measurement using multiclass SVM and KNN classifier" 26 April 2019

[12] Elham Khalili, Samaneh Kouchaki, Shahin Ramazi, Faezeh Ghanati "Machine Learning Techniques for Soybean Charcoal Rot Disease Prediction" 14 December 2020

[13] Miao Yu, Xiaodan Ma, Haiou Guan, Meng Liu, Tao Zhang "A Recognition Method of Soybean Leaf Diseases Based on an Improved Deep Learning Model" 31 May 2022

[14] Munirah Hayati Hamidon, Tofael Ahamed "Detection of Tip-Burn Stress on Lettuce Grown in an Indoor Environment Using Deep Learning Algorithms" 24 September 2022

[15] Kavir Osorio, Andrés Puerto, Cesar Pedraza, David Jamaica, Leonardo Rodríguez "A Deep Learning Approach for Weed Detection in Lettuce Crops Using Multispectral Images" 28 August 2020

[16] J. Amara, B. Bouaziz, and A. Algergawy, "A Deep Learning-based Approach for Banana Leaf Diseases Classification" in B. Bernhard Mitschang, Norbert Ritter, Holger Schwarz, Meike Klettke, Andreas Thor, Oliver Kopp, Matthias Wieland (Hrsg.): BTW 2017 – Workshopband, Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 79.

[17] W. Liao, D. Ochoa, L. Gao, B. Zhang, and W. Philips, "Morphological Analysis for banana disease detection in close range hyperspectral remote sensing images," in Proceedings of the IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium, pp. 3697–3700, Yokohama, Japan, 28 July-2 August 2019.

[18] S. kaur, G. Babbar, and Gagandeep, "Image processing and classification, A method for plant disease detecion," International Journal of Innovative Technology and Exploring Engineering, vol. 8, no. 9S, 2019.

[19] Shimamura, S.; Uehara, K.; Koakutsu, S. Automatic Identification of Plant Physiological Disorders in Plant Factory Crops. IEEJ Trans. Electron. Inf. Syst. 2019, 139, 818–819.

[20] S. Mishra, R. Sachan, and D. Rajpal, “Deep convolutional neural network based detection system for real-time corn plant disease recognition,” Procedia Computer Science, vol. 167, pp. 2003–2010, 2020.

[21] D. A. Kumar, P. S. Chakravarthi, and K. S. Babu, “Multiclass Support Vector Machine Based Plant Leaf Diseases Identification from Color, Texture and Shape Features,” in Proceedings of the 2020 ;ird International Conference on Smart Systems and Inventive Technology (ICSSIT), pp. 1220–1226, IEEE, Tirunelveli, India, August 2020.

[22] V. Mazzia, A. Khaliq, and M. Chiaberge “Improvement in land cover and crop classification based on temporal features learning from sentinel-2 data using recurrent-convolutional neural network (R-CNN)”

[23] N. Çetin, K. Karaman, E. Beyzi, C. Sa˘glam, and B. Demirel, “Comparative evaluation of some quality characteristics of sunflower oilseeds (helianthus annuus L.) through machine learning classifiers,” Food Analytical Methods, vol. 14, no. 8, pp. 1666–1681, 2021.

[24] Sharif M, Khan MA, Iqbal Z, Azam MF, Lali MIU, Javed MY. Detection and classification of citrus diseases in agriculture based on optimized weighted segmentation and feature selection. Comput Electron Agric 2018;150:220–34. https://doi.org/10.1016/j.compag.2018.04.023.

[25] Lu J, Ehsani R, Shi Y, Abdulridha J, de Castro AI, Xu Y. Field detection of anthracnose crown rot in strawberry using spectroscopy technology. Comput Electron Agric 2017.

[26] Ruben Van De Vijver, Koen Mertens, Kurt Heungens, Ben Somers, David Nuyttens, Irene Borra-Serrano, Peter Lootens, Isabel Roldán-Ruiz, Jürgen Vangeyte, Wouter Saeys, In-field detection of Alternaria solani in potato crops using hyperspectral imaging, Computers and Electronics in Agriculture, Volume 168, 2020, 105106, ISSN 0168-1699, https://doi.org/10.1016/j.compag.2019.105106.

[27] D. Zhang, Y. Zhang, Q. Li, T. Plummer and D. Wang, "CrowdLearn: A Crowd-AI Hybrid System for Deep Learning-based Damage Assessment Applications," 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), Dallas, TX, USA, 2019, pp. 1221-1232, doi: 10.1109/ICDCS.2019.00123.

[28] Noyan, Mehmet Alican. "Uncovering bias in the PlantVillage dataset." arXiv preprint arXiv:2206.04374 (2022).

[29] A. Mikołajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," 2018 International Interdisciplinary PhD Workshop (IIPhDW), 2018, pp. 117-122, doi: 10.1109/IIPHDW.2018.8388338.

[30] Medhi, Epsita, and Nabamita Deb. “PSFD-Musa: A Dataset of Banana Plant, Stem, Fruit, Leaf, and Disease.” Data in Brief 43 (2022): 108427.

[31] A. Mikołajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," 2018 International Interdisciplinary PhD Workshop (IIPhDW), 2018, pp. 117-122, doi: 10.1109/IIPHDW.2018.8388338.

[32] Maraghehmoghaddam, Armin. “Synthetic Data Generation for Deep Learning Model Training to Understand Livestock Behavior,” n.d. https://doi.org/10.31274/etd-20200902-98

[33] Pratiwi, Heny, Agus Perdana Windarto, S. Susliansyah, Ririn Restu Aria, Susi Susilowati, Luci Kanti Rahayu, Yuni Fitriani, Agustiena Merdekawati, and Indra Riyana Rahadjeng. “Sigmoid Activation Function in Selecting the Best Model of Artificial Neural Networks.” Journal of Physics: Conference Series 1471, no. 1 (2020): 012010.

[34] Namin, Ashkan & Leboeuf, Karl & Muscedere, Roberto & Wu, Huapeng & Ahmadi, Majid. (2009). Efficient hardware implementation of the hyperbolic tangent sigmoid function. Proceedings - IEEE International Symposium on Circuits and Systems. 2117 - 2120. 10.1109/ISCAS.2009.5118213.

[35] Bodyanskiy, Yevgeniy, Anastasiia Deineko, Viktoria Skorik, and Filip Brodetskyi. “Deep Neural Network with Adaptive Parametric Rectified Linear Units and Its Fast Learning.” International Journal of Computing, 2022, 11–18.

[36] Abien Fred Agarap. (2018). Deep Learning using Rectified Linear Units (ReLU).https://doi.org/10.48550/arXiv.1803.08375

[37] I. Kouretas and V. Paliouras, "Simplified Hardware Implementation of the Softmax Activation Function," 2019 8th International Conference on Modern Circuits and Systems Technologies (MOCAST), Thessaloniki, Greece, 2019.

[38] Zhuang Li, Hanzi Mao, Chao-Yaun, Christoph Feichtenhofer, Trevor Darrell and Saining Xie, “A ConNet for the 2020s” , 2020.

[39] Menghani, Gaurav. “Efficient Deep Learning: A Survey on Making Deep Learning Models Smaller, Faster, and Better.” ACM Computing Surveys 55, no. 12 (2023): 1–37.

[40] Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747..

[41] Bottou L. (2012) Stochastic Gradient Descent Tricks. In: Montavon G., Orr G.B., Müller KR. (eds) Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, vol 7700. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-35289-8_25

[42] Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. Journal of Machine Learning Research, 12, 2121–2159.

[43] Zhang, Rui, Weiguo Gong, Victor Grzeda, Andrew Yaworski, and Michael Greenspan. “An Adaptive Learning Rate Method for Improving Adaptability of Background Models.” IEEE Signal Processing Letters 20, no. 12 (2013): 1266–69. https://doi.org/10.1109/lsp.2013.2288579.

[44] Peto, Levente, and Janos Botzheim. “Parameter Optimization of Deep Learning Models by Evolutionary Algorithms.” 2019 IEEE International Work Conference on Bioinspired Intelligence (IWOBI), 2019.

[45] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. 2014. arXiv:1412.6980v9 (2014)

[46] “Follow-the-regularised-leader and Mirror Descent” (2020) Bandit Algorithms, pp. 286–305. Available at: https://doi.org/10.1017/9781108571401.035.

[47] ALHAWAS, Nagham, and Zekeriya TÜFEKCİ. “The Effectiveness of Transfer Learning and Fine-Tuning Approach for Automated Mango Variety Classification.” European Journal of Science and Technology, European Journal of Science and Technology, Mar. 2022. Crossref, https://doi.org/10.31590/ejosat.1082217.

[48] Kundjanasith Thonglek, Keichi Takahashi, Kohei Ichikawa, Chawanat Nakasan, Hidemoto Nakada, Ryousei Takano, Hajimu Iida, "Retraining Quantized Neural Network Models with Unlabeled Data," 2020 International Joint Conference on Neural Networks (IJCNN), 2020, pp. 1-8, doi: 10.1109/IJCNN48605.2020.9207190.

[49] A. Ghasemian, H. Hosseinmardi and A. Clauset, "Evaluating Overfit and Underfit in Models of Network Community Structure," in IEEE Transactions on Knowledge and Data Engineering, vol. 32, no. 9, pp. 1722-1735, 1 Sept. 2020, doi: 10.1109/TKDE.2019.2911585.

[50] S. K. Noon, M. Amjad, M. A. Qureshi and A. Mannan, "Overfitting Mitigation Analysis in Deep Learning Models for Plant Leaf Disease Recognition," 2020 IEEE 23rd International Multitopic Conference (INMIC), 2020, pp. 1-5.

[51] Keith, Michael. “Ridge and Lasso.” Machine Learning with Regression in Python, 2020. https://doi.org/10.1007/978-1-842-6583-3_4.

[52] K. P. Asha Rani and S. Gowrishankar, "Pathogen-Based Classification of Plant Diseases: A Deep Transfer Learning Approach for Intelligent Support Systems," in IEEE Access, vol. 11, pp. 64476-64493, 2023, doi: 10.1109/ACCESS.2023.3284680.