applied sciences

MDPI

Article

Sesame Plant Disease Classification Using Deep Convolution Neural Networks

Eyerusalem Alebachew Nibret [1,2,*], Azanu Mirolgn Mequanenit [1,3], Aleka Melese Ayalew [1,4], Kusrini Kusrini [5] and Rodrigo Martínez-Béjar [3]

[1] Department of Information Technology, University of Gondar, Gondar 196, Ethiopia; azanumirolgn06@gmail.com (A.M.M.); melese1820@gmail.com (A.M.A.)
[2] Department Electrical Engineering and Industrial Informatics, Federal University of Parana, Curtiba 80060-000, Brazil
[3] Department of Computer Science, University of Murcia, 30100 Murcia, Spain; rodrigo@um.es
[4] Center for Machine Vision and Signal Analysis, University of Oulu, 90014 Oulu, Finland
[5] Department of Computer Science, University AMIKOM Yogyakarta, Yogyakarta 55281, Indonesia
* Correspondence: eyualebachew@gmail.com

Abstract: Monitoring sesame plant health and detecting disease early are essential to reducing disease spread and facilitate effective management practices. In this research, we developed an image classification model to detect bacterial blight-infected, phyllody-infected, and healthy sesame crops. Since images were necessary to carry out this study, we collected 2300 images at the Gondar and Humera Agriculture Research Centers and directly from the field in Metema. Since the collected images were limited, to increase the number of images in the dataset, we used image augmentation with different variations. In the image preprocessing step, we used a median filter for noise filtering, and contrast stretching techniques were used for image contrast and brightness enhancement. SegNet semantic segmentation, which is deep convolution neural network-based architecture, was used to segment the leaf part of the image from the background. In the feature extraction and classification steps, a deep convolutional neural network was used. Finally, we evaluated the proposed model and compared it with two recent deep convolution neural network models, namely, Xception and InceptionV3. The proposed model for the classification of sesame diseases achieved better accuracy, with 96.67% testing accuracy, 97.78% validation accuracy, and 98% training accuracy.

Keywords: sesame plant diseases; deep convolution neural networks; image enhancement techniques; SegNet semantic segmentation

Academic Editor: Andrea Prati

Received: 20 December 2024
Revised: 5 February 2025
Accepted: 11 February 2025
Published: 17 February 2025

Citation: Nibret, E.A.; Mequanenit, A.M.; Ayalew, A.M.; Kusrini, K.; Martínez-Béjar, R. Sesame Plant Disease Classification Using Deep Convolution Neural Networks. *Appl. Sci.* **2025**, *15*, 2124. https://doi.org/10.3390/app15042124

1. Introduction

Sesame (*Sesamumindicum* L. Pedaliaceae) is one of the oldest flowering oil crops and is thought to have originated in Africa. Sesame seed oil possesses excellent nutritional, medicinal, cosmetic, and culinary qualities [1]. In Africa, various wild forms occur, as well as a smaller number in India. It is widely naturalized worldwide in tropical and subtropical regions [2]. According to the global sesame seed market, sesame production in 2016 was 6.1 million tons, with Tanzania, Myanmar, India, and Sudan as the largest producers [3]. Around the globe, the consumption of sesame is increasing year by year, and the market trend is also increasing. The production, growth, and quality improvement of oilseeds can substantially contribute to economic development at the national, regional, and family levels [4,5]. Bacterial blight, leaf spot, phyllody, and wilting are the major diseases

of sesame plants [6]. Such diseases have major impacts on the production of sesame, including reductions in seed quality, yield, oil content, and plant health. Since disease decreases the quality of seeds as well as the content of the product, income from exports and local sales decrease. It affects the economy of the country, including everyone who directly or indirectly benefits from sesame cultivation [7]. Monitoring sesame plant health and detecting the disease early is essential to facilitate effective management practices and reduce disease spread [8]. The identification of sesame diseases by observation is time-consuming, tedious, and requires human labor, and this may lead to incorrect disease identification and also the inappropriate usage of pesticides [8]. This traditional method is limited due to the low capacity of human labor to cover large areas and the lack of local experts, and so on. Hence, it is required to develop computational methods that make the process of disease detection and classification easier [9]. To this end, researchers in [10] presented the CLIPS expert system, which is used to identify sesame diseases. But the developed system requires careful identification of the symptoms by users to identify the disease correctly. Also, the system is only capable of doing what it is designed for and what the user has selected. It cannot provide creative responses for different cases.

For the detection of crop diseases, different techniques are used, such as machine learning [11]. Machine learning is a sub-area of artificial intelligence that is able to find solutions to problems independently by identifying patterns in databases [12]. The types and symptoms of diseases that commonly affect different plants can be different. The literature has shown that since crops' morphology, color, and texture features are different, a model proposed for one agricultural crop cannot be directly applied to others [9].

In this study, a model is developed to detect diseases in sesame crop of three classes; bacterial blight-infected, phyllody-infected, and healthy sesame crops. Some sesame diseases show no clear symptoms and exhibit a wide range of colors in healthy leaves. This makes it difficult to detect sesame diseases. As a result, to improve image quality and recognition accuracy, we use median filtering to remove noise, and contrast and brightness enhancement are performed using a contrast stretch technique. The SegNet semantic segmentation technique is used to segment leaves from unwanted backgrounds. Feature extraction and classification are performed using a deep convolution neural network architecture with improved layers and parameters.

The remaining parts of this paper are organized as follows. Section 2 presents a review of related works. Section 3 describes the methodology used in the development of the proposed sesame disease classifier, while the results and discussion are pointed out in Section 4. Finally, some conclusions are put forward in Section 5.

2. Related Work

Researchers in [10] proposed a rule-based expert system for diagnosing and treating sesame plant disease. The system aimed to help agricultural experts identify the four common sesame plant diseases, which are phyllody, dry root rot, Phytophthora blight, and Alternaria blight. To develop and implement the expert system, they used the C Language Integrated Production System (CLIPS) expert system and Delphi language. This system determines the diagnosis by asking the users to choose the symptoms that appear in the sesame plants. Based on the selected symptoms, the system provides a recommendation of the disease to the users. The authors obtained encouraging results.

Authors in [11] presented a study to identify leaf rust disease in coffee plants, which is caused by pathogenic fungi that attacks the underside of coffee leaves. To train and evaluate their model for rust detection, they prepared a dataset that contained 159 images of coffee leaves. Their proposed architecture contains only two convolutional layers followed by a nonlinear ReLU filter and a max pooling layer. Their approach can detect the infected

area and uses morphological erosion to improve the detection for high precision. Their model achieved 95% accuracy.

In [13], as the researchers discussed, they developed model to detect mango plant diseases. For the identification of the diseases, they used a dataset that consisted of 8853 total mango leaf images. The images they collected consisted of four diseases such as powdery mildew, anthracnose, red rust, and golmich. In their proposed model, they retrained the ResNet50 convolution neural network system. They measured the performance of the training model across the three architectures of ResNet, which are ResNet18, ResNet34, and ResNet50. ResNet50 has 50 deep layers, but its computational time cost is only 2.5 times higher than that of the other two. Based on their experiment using the mango leaf dataset, the ResNet18 CNN, ResNet34 CNN, and ResNet50 architecture achieved 91%, 90.88%, and 91.50% accuracy, respectively. The authors concluded that the deep convolution neural network is a powerful technique to detect mango leaf diseases in the field with high precision.

In [14], researchers proposed a maize leaf disease identification technique using the GoogleNet and Cifar10 convolution neural network models. They collected 500 maize leaf images with similar backgrounds from different Google websites and cleaned the images using developed Python scripts. In their image processing steps to enhance the images, they performed size and format normalization using Python scripts based on the OpenCV framework automatically. Based on the experimental result, they recorded an accuracy of 98.5% and 97.1% using the original GoogleNet and Cifar10 models, respectively, and a high accuracy of 98.9% and 98.8% from the improved GoogleNet and Cifar10 models, respectively. The researchers concluded that reasonably improving the models allows for a reduction in the convergence iteration and leads to an increase in the identification accuracy.

In [15], researchers proposed a model for detecting three rice plant diseases and healthy rice plants using deep convolutional neural networks. They used 619 rice plant images from the field belonging to four classes. The number of images used for each class was 88 for leaf brown spot-infected plants, 240 for bacterial leaf blight-infected plants, 100 for sheath blight-infected plants, and 191 for healthy leaves. We trained and tested the model using various partitioning strategies: 60% of the data for training, 40% for testing, 70% for training, 30% for testing, and 80% for training, and 20% for testing. They used AlexNet to extract features and a support vector machine (SVM) to classify them. Using the above training–testing partitions, they achieved 91.37% with the training–testing partition of 80–20%.

In [16], researchers proposed two convolution neural network models that detect blackleg disease in potato plants. Two deep convolutional neural networks for healthy and diseased plants, ResNet18 and ResNet50, were trained on RGB images. For model training and testing, the authors used 532 images collected in the field. In the training process, they used potato images that were isolated and did not overlap with each other or other plants. For this reason, to obtain the isolated images, they only collected the images during the first weeks after plant emergence. The computation times for training and testing their model were four hours and 6 min for ResNet18 and seven hours and 5 min for ResNet50. ResNet18 achieved higher performance, with a precision of 95% and recall of 91% for the disease class. However, the proposed model was not able to deal with overlapping plants. Also, the time it took for training and testing was extensively high. In addition, validation of the model, which allows for the optimization of performance, was not implemented. Many researchers have developed machine learning models for detecting and classifying numerous crop diseases using different techniques. However, the previous works have some limitations, such as a lack of creative response, no common sense used in decision making, program unable to processing high-resolution images, using a small number of images with similar

backgrounds, class imbalance in the manipulation of data, high computational time, and not being able to deal with overlapping plant images.

To overcome the limitations described above as described in above, the proposed approach in this study utilized a range of techniques. Utilizing deep convolutional neural networks allowed the model to learn relevant features and patterns from the input images while including image preprocessing steps, such as median filtering and contrast enhancement, which helped to standardize low-quality samples. With SegNet semantic segmentation (specifically trained with our domain-agnostic examples of leaf images due to the existing classes), we are able to achieve the separation between object-of-interest and background, which helped in overcoming the overlapping plant image problem, as the model focuses only on the region where the target/leaf is present and on its attributes that differentiate it from the non-background area behind. This increases the diversity of training data distinguishably, mitigating the class imbalance problem and helping to better capture alternate/random images. In addition, carefully designing the CNN architecture with optimal layers and parameters resulted in a model that contains rather few parameters as compared to some other typical architectures. It also has high performance due to this additional resource consumption aspect, which overcomes one key limitation of previous attempts, i.e., very high computational time.

3. Methodology

3.1. System Architecture

The major steps of the proposed system architecture are shown in Figure 1. The steps include image preprocessing, image segmentation, feature extraction, and sesame crop disease classification. In the image processing step, we resized the images to a standard size and employed filtering techniques for noise removal and image enhancement. In addition, image segmentation was performed to identify the region of interest and distinguish the foreground image from the background image. The CNN technique was used to extract features from the sesame image. In the feature extraction and classification steps, a deep convolution neural network was used. The architecture illustrates the overall process followed to train, validate, and test the proposed model.

Figure 1 depicts the data collection and preparation pipeline of a deep-learning image classification problem. The following mathematical representation covers the step-by-step transformations of the input image through the preprocessing, feature extraction, and classification pipeline of the proposed CNN model.

The 2D matrix input image representation:

$$I(x, y) \in R^{M \times N \times C} \tag{1}$$

where M and N are the spatial dimensions (height and width) of the image, and C represents the number of RGB channels.

Preprocessing and segmentation: image preprocessing techniques such as normalization, contrast enhancement, and noise reduction can be represented mathematically as

$$I'(x, y) = \frac{I(x, y) - \mu}{\sigma} \tag{2}$$

where μ and σ are the mean and standard deviation of pixel intensities. Segmentation may involve thresholding:

$$S(x, y) = \begin{cases} 1, \ if \ I'(x, y) \geq T \\ 0, \ otherwise \end{cases} \tag{3}$$

where T is a threshold value.

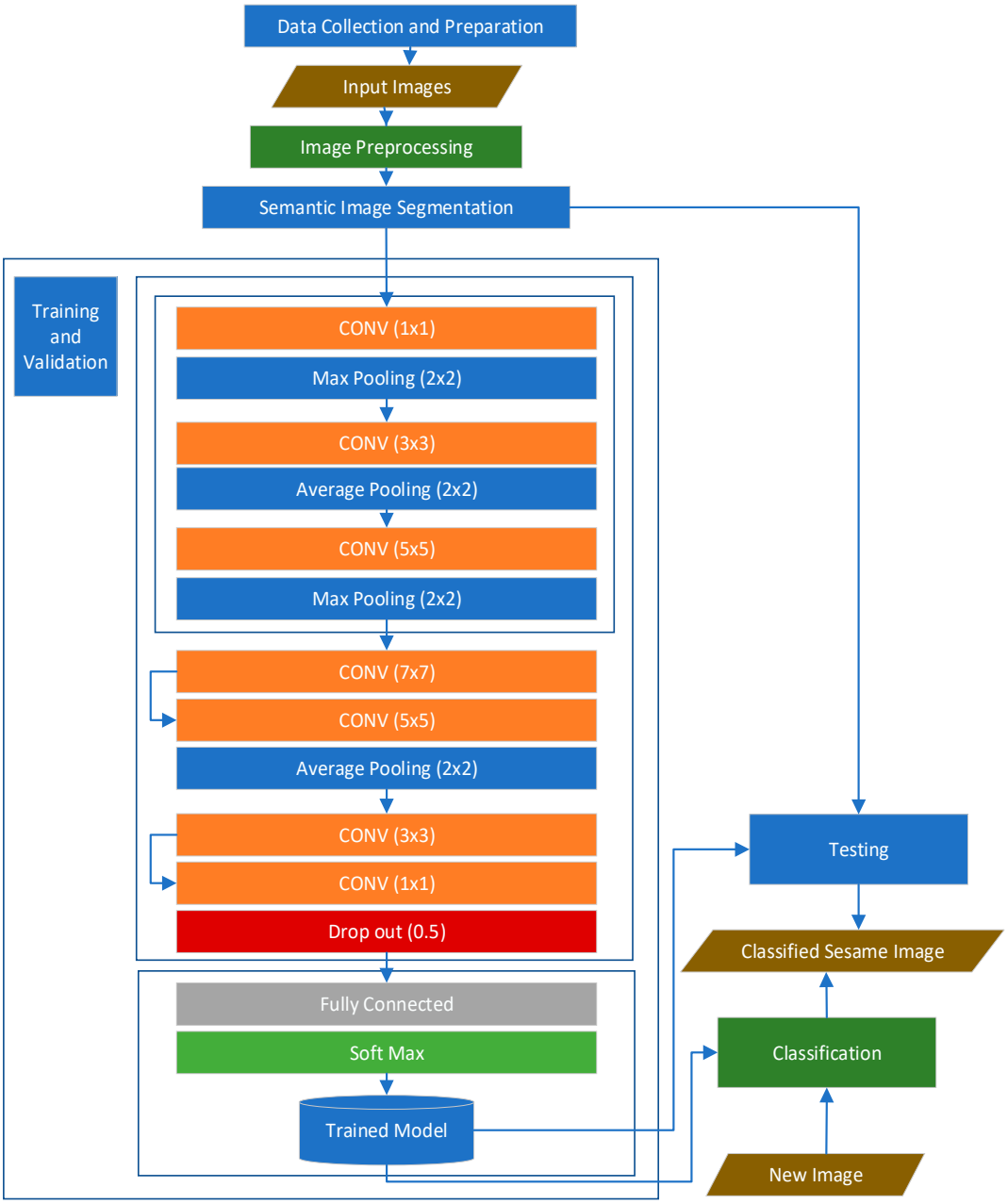


Figure 1. Shows the proposed system architecture.

Convolutional layers apply a filter W (of size $k \times k$ times) to the image:

$$F_{ij}^{l} = \sum_{m=0}^{K-1} \sum_{n=0}^{K-1} W_{mn}^{(l)} I''(i-m, j-n) b^{l} \tag{4}$$

where $W^{(l)}$ is the weight of the n^{-th} layer, $b^{(l)}$ is the bias, and k is the filter size.

In the pooling layers, max pooling and average pooling reduce spatial dimensions.

Max pooling:

$$Pmax(i,j) = \max_{(m,n) \in R} F(i+m, j+n) \tag{5}$$

Average pooling:

$$Pavg(I,j) = \frac{1}{|R|} + \sum_{(m,n \in R)} F(i+m, j+n) \tag{6}$$

where R is the pooling window size (2×2).

Dropout regularization is applied to prevent overfitting:

$$O = D \cdot F \tag{7}$$

where $D \sim$ Bernoulli(p) (binary mask with probability p).

The extracted features are flattened into a vector and fed into a fully connected layer.

$$Z^{(L)} = W^{(L)} P^{(L-1)} + b^{(L)} \tag{8}$$

The final classification is performed using the softmax function:

$$P(yi) = \frac{e^{zj}}{\sum_j e^{zj}} \tag{9}$$

where z_i represents the logits (outputs before activation) for class i.

The trained model assigns a label to the input sesame leaf image based on the highest probability output from softmax.

3.2. Data Collection and Preparation

In this research, 2300 sesame leaf images were collected from the Humera Agricultural Research Center (HARC) and Gondar Agriculture Research Center (GARC), and most of the images were collected directly from a sesame field in Metema with the help of experts. These collections allowed the model to train with different image properties and conditions, which allowed us to obtain better and more consistent results. Since our dataset had a limited number of images, we used data augmentation techniques, i.e., rotation, flipping, stretching, translation, and shearing, to increase the number of images. For developing and testing the proposed model, we used a total of 9200 augmented images. The dataset was divided into training, validation, and test sets.

Sesame sample images: The sampled sesame images, which were collected from the different sites mentioned above to develop the predictive model, are shown in Figure 2. The images are representative of the classes in the proposed model.

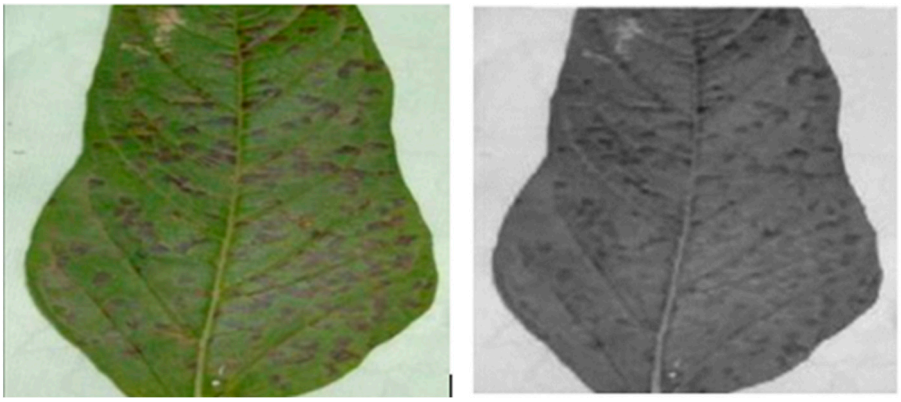
(**a**) (**b**) (**c**)

Figure 2. (**a**) Bacterial blight-infected, (**b**) phyllody-infected, and (**c**) healthy sesame leaves.

3.3. Image Preprocessing

Here, in the first step of our experiment, the input images are converted into grayscale images using the rgb2grayscale() Matlab function. The converted images are adjusted to a size of 299 by 299 using Matlab's built-in scripts. In this experiment, the preprocessing

step includes adjusting the size and color, removing the noise, which may be dust, water droplets, or other particles since the images are field images, and enhancing the contrast and brightness of the images. In Figure 3, the RGB label image and its converted grayscale label image are shown. The following steps are used in the algorithm for grayscale conversion:



(**a**) (**b**)

Figure 3. (**a**) RGB label image; (**b**) grayscale label image.

3.3.1. Contrast Stretching

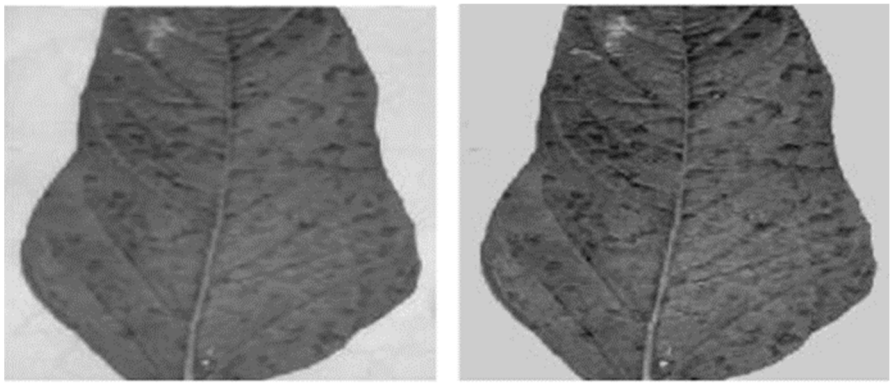
Contrast Stretching was applied to the collected data to enhance the image quality. Image contrast is a measure of the spread of its histogram. Contrast stretching aims to expand the dynamic range of an image. It transforms the grayscale level in the range {0, 1, …, $L - 1$} using a piecewise leaner function [17] to determine the limits over which image intensity values will be extended. These lower and upper limits will be called a and b, respectively (for standard 8-bit grayscale pictures, these limits are usually 0 and 255). The histogram of the original image is examined to determine the value limits (lower = c, upper = d) in the unmodified image. If the original range covers the full possible set of values, straightforward contrast stretching will achieve nothing, but even then, sometimes, most of the image data are contained within a restricted range; this restricted range can be stretched linearly, with original values that lie outside the range being set to the appropriate limit of the extended output range. Then, for each pixel, the original value r is mapped to the output value s using the appropriate function. The equation for the original value r is mapped to the output value s for each pixel as follows:

$$S = (r - c)\left(\frac{b - a}{d - c}\right) + a \tag{10}$$

3.3.2. Median Filtering

Median filtering was applied to retain the important data in images [18]. Figure 4 shows the original grayscale label image with respect to its preprocessed result. In the following equation, Yi represents the filtered output sequence of X, where n is the number of elements in the input sequence X, Ji is a subset of the input sequence X centered on the ith element of X, and the indexed elements outside the range of X equal zero. The following equation describes J_i.

$$Yi = Median(Ji),\ for\ i = 0, 1, 2, \ldots, n - 1 \tag{11}$$



(a) (b)

Figure 4. (**a**) Original grayscale label image; (**b**) median filtered and contrast-enhanced image.

A more advanced alternative that was not considered in this study is median-recursive filtering. Median-recursive filtering combines impulse noise resistance with better preservation of useful signals, which could lead to more accurate reconstructions in high-noise conditions [19]. Recursive filters generally provide a more consistent smoothing effect without excessive blurring of critical image details. Future research could explore the integration of recursive or median-recursive filtering techniques to further improve disease classification accuracy by better preserving essential structures within the images.

The preprocessing technique's performance may differ from image to image due to the problems to be solved in the images. So, we compared techniques, i.e., histogram equalization (HE), contrast limited locally adaptive histogram equalization (CLAHE), brightness-preserving bi-histogram equalization, and contrast stretch, using randomly selected images. The comparison was performed using the peak signal-to-noise ratio (PSNR), structural similarity value (SSIM), and mean square error (MSE), which were to find the quality of the image after the removal of noise. The three techniques other than contrast stretch yielded small values of SSIM and PSNR and high values of MSE. This means that they hardly improved the contrast of the image. This is because HE only focuses on the global contrast of the image, leading to the loss of local details. Also, in CLAHE, vagueness is introduced to an image during digitalization in the form of imprecise boundaries and color values due to weak and non-uniform lightning in the imaging system. The contrast stretching method yields better SSIM and PSNR values and lower MSE values compared to those of the other methods. Contrast stretching is all about increasing the difference between the maximum intensity value in an image and the minimum value.

3.4. Image Segmentation

In this study, we used the SegNet semantic image segmentation network, which is a modern and effective deep learning-based segmentation technique. It is a flexible, general approach and the current state-of-the-art for image segmentation [20]. This network can learn the data and be trained on those data to identify patterns and segment the important parts [21]. Here, there are reasons that lead us to use a modern CNN segmentation technique that learns and segments different images based on their characteristics. The first reason is that some types of sesame disease may reduce the size and change the structure of the leaves. The other is that since the images are field images, the color and intensity of the background, which is the surrounding field, are similar to those of the leaf part. Semantic segmentation (SegNet) is a CNN segmentation technique that associates each pixel of an image with a class label. It is currently utilized as a highly flexible technique. It has the ability to learn new features.

Segmentation is performed using SegNet, a deep convolutional encoder-decoder network. The segmentation output S is obtained as follows:

$$S(x,y) = arg\ \underset{k}{max} \boldsymbol{P}(C_k \mid \boldsymbol{I''}) \quad (12)$$

where $P(C_k|I'')$ is the probability of class C_k (leaf or background) for each pixel.

To use a SegNet semantic segmentation network in our experiment, we followed these steps:

1. Create pixel label images of the original sesame leaf images using the ground truth labeler;
2. Load the training data and create a pixel label data store and an original image data store;
3. Add the pre-trained VGG16 CNN model;
4. Load the pre-trained SegNet semantic segmentation network;
5. Train the SegNet network with our dataset images;
6. Display and store the segmentation results.

After training the SegNet model using pixel-labeled sesame leaf images with the original sesame images, the trained network segmented the input images. Figure 5 shows the sample segmentation process and results.

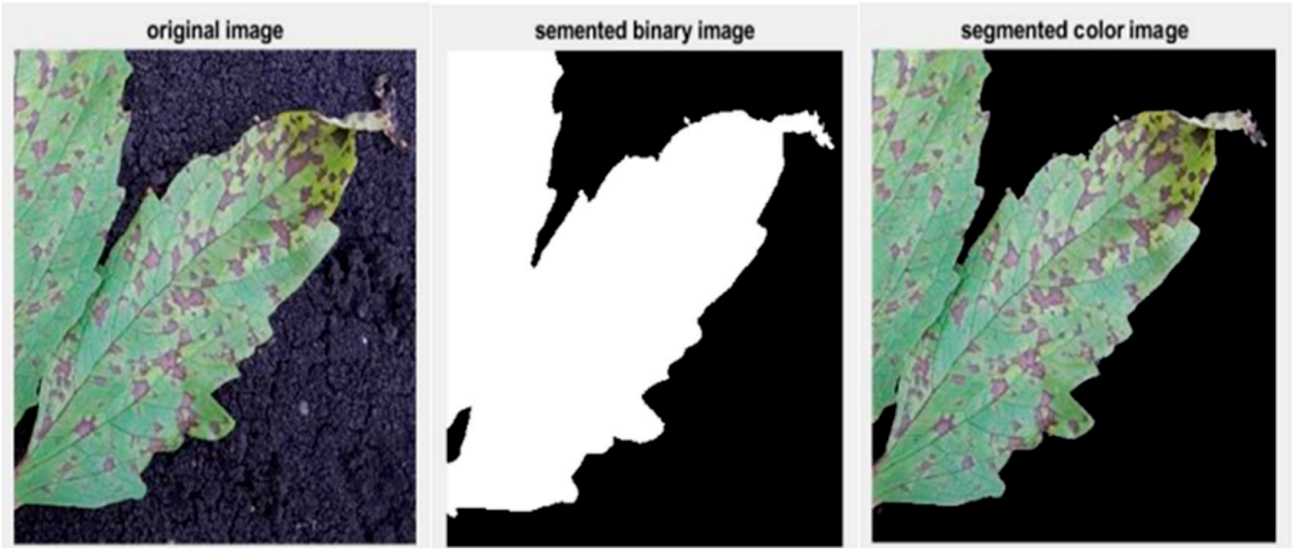


Figure 5. Sample image segmentation results.

3.5. Image Augmentation

In convolution, neural network data quantity limitation is a major problem in the predictive model performance. Also, imbalanced classes can prevent the model from understanding and treating the classes properly [22]. To solve this problem using a convolution neural network, image augmentation is the best and most useful method. Image augmentation is used to artificially increase the number of images in the dataset. It creates many forms of images by applying different kinds of variation to a single image. This allows the model to learn and understand features from many image variants and leads the model to provide good prediction quality. The augmentation technique is applied to the training set to effectively train the model. The testing set is used to test the efficiency of the model, so artificially duplicating the image is not useful here [22].

Image augmentation techniques that we used in our experiment are presented below.

Rotation is an augmentation technique that duplicates the images by rotating them from the value zero up to 360 degrees. To obtain the correct rotation result, rotation must be performed with uniform random generation. In our experiment, we used random rotation within the range of the images in the range of [−45, 45].

Flipping is a technique that mirrors the images with uniform random generation values.

Stretching is an augmentation technique used to create a scale transformation that uses a scale factor chosen randomly in the [1.2, 1.5] range to resize the input image. In the horizontal and vertical directions, this transformation resizes the image by the same element.

Translation creates a translation transformation that shifts the input image horizontally and vertically by a distance selected randomly in the range of [−50, 50] pixels.

Shearing is used to create a shear transformation, with the shear angle selected randomly in the range of [−30, 30].

3.6. Feature Extraction and Classification

The feature extraction process in our study extracts meaningful information from the input sesame leaf images. The proposed models perform feature extraction and classification using a convolutional neural network. In recent times, deep convolution neural network techniques have exhibited very high performance in different areas, such as image classification and other fields [23]. The convolution layer has the purpose of extracting features from the inputs, i.e., horizontal edges, vertical edges, pixel values, and so on. It learns features at different levels of abstraction as the layer increases. Based on the extracted features, the fully connected layer performs classification into different classes, and the results are transferred through the output layer (Softmax layer) [24].

The structure of the proposed model training, validation, and testing phases are presented below. Since the proposed model was developed based on a deep convolutional neural network, it consisted of several layers that represented different building blocks of the convolutional neural network. The layers and parameters that we applied in the proposed model to learn the features predict the classes based on the extracted features, and produce a classification result as the output are discussed below.

Input Layer: We used an input size of 299 × 299 × 3. We selected the input size by referring to the performance of the proposed models using different image sizes. This input size allowed our model to perform well since large patterns in the image are more visual. In addition, we used this size to compare model performance with that of recent models that have similar input sizes.

Convolution Layer: The proposed model consisted of 10 convolution layers with different characteristics and parameters. In the convolution layer, there are five parameters. The first parameter, which is filter size, is one of the main parameters that allows the model to detect the features. As the filter size increases, the detection rate also rises since the filters can cover larger pixels and learn features from those pixels simultaneously. The layer learns the features by localizing the regions and scanning through an image based on the filter size. Therefore, we specified the proper region to be localized at a given time using this parameter. In the proposed model, we used one by 1, 3 by 3, 5 by 5, and 7 by seven as filter sizes. The number of filters, which is the second parameter, refers to the neurons in the output of the convolutional layer. In the proposed model, we used 8, 16, 32, 64, 96, 128, 160, and 192 filters. As we moved down the network, we increased the number of filters since an increased number of neurons provides more strength and capability to the proposed network. The third parameter is stride. We used one stride value that shifted the pixels throughout the network. Since we did not want to reduce the size of the output in the convolution layer, we used a padding value of "same", which implies that the size of

the output is equal to the input size with a stride value of 1. This was used to control the output size of the layer.

The parameters are selected based on the cases that we classified images into. In bacterial blight-infected leaves, the symptoms are small spots that are distributed over the leaf, and each spot can cover around 7 to 9 pixels in the image. In contrast, the phyllody-infected leaves seem healthy when we look closer because the symptoms are different, changing the structure or size of the leaf and reducing internode distance; the symptoms may cover 14 to 25 pixels. Therefore, for such cases, we selected 1×1 to 7×7 filter sizes since we agreed that such sizes could cover the pixels of interest and learn features from such pixels at the same time. The other parameters were also selected based on our dataset.

Batch Normalization Layer: We used this layer between convolutional layers and ReLU activation layers in all convolution operations to normalize activation and speed up the training of the proposed model. While ReLU is widely used due to its simplicity and efficiency, it has a known drawback: it can lead to dead neurons when the weights are updated such that input values remain permanently negative, preventing further learning by those neurons. To address this issue, modified activation functions such as SmoothReLU can be considered. SmoothReLU offers a continuous and differentiable alternative to ReLU, mitigating the problem of dead neurons and potentially improving the learning capacity of the network [25]. Future work could explore the integration of SmoothReLU to enhance model performance and stability.

Activation Layer: We used the ReLU activation function for all convolution operations. This layer enables the network to learn complex patterns from the inputs. The ReLU function replaces the 0 and negative outputs of the neurons with 0, which deactivates the neuron, and when the output is positive, it leaves the neuron as is. This implies that the function does not activate all the neurons at the same time. Due to this, it computes faster than other activation functions, and this makes a significant difference in training time.

Pooling Layer: We used mixed pooling with max pooling and average pooling methods. The max pooling layer brightens pixels, while average pooling smooths out the images and the sharp features that may not be identified. Since the images we used in our experiment did not have a constant character, to maximize the benefits of the two pooling methods, we applied both of them. Since the size of the input image is large, we used a pooling size of 2 and stride of 2 at the end of every convolution operation to reduce the spatial size. A pooling size of 2 by 2 covers 4 values and converts them to a single value to down sample the size of the output and remove redundant information.

Dropout Layer: We used a dropout layer at the end of the convolution layer. This prevents the proposed model from overfitting since this layer effectively changes the underlying network architecture between iterations by randomly sets input elements to zero with a given probability. A higher number results in more elements being dropped during training.

Fully Connected Layer: In this layer, the features that are learned in the convolution process are combined to identify the pattern. We used a single fully connected layer with an output size of 3, which is the number of classes that we identified with the proposed model. The identified pattern is used to classify the images. This is why we used an output size parameter similar to the number of classes in our dataset.

Softmax Layer: This takes numbers from all three neurons of a fully connected layer and sums them to a single one that provides the final output corresponding to the particular class. Then, this can be used to determine classification probabilities for each classification output.

In the model training process, we used 40 epochs, which implies that the training process moved 40 times over the training data to extract features and train the model. We

also adjusted other parameters concerning the number of images that we used to train and the problem domain that we solved using the model.

3.7. Neural Network Training Algorithm

The neural network training process in this study is based on a supervised learning approach using backpropagation and stochastic gradient descent (SGD) optimization. Below is the detailed mathematical formulation of the training algorithm.

Forward Propagation: Each input image **X** is passed through a sequence of layers, including convolutional, activation, pooling, and fully connected layers. The transformation at each layer is **l** defined as follows:

$$Z^{(L)} = W^{(L)} P^{(L-1)} + b^{(L)} \tag{13}$$

where $W^{(L)}$ is the weight matrix of the layer, $P^{(L-1)}$ is the activation from the previous layer, and $b^{(L)}$ is the bias vector of the layer.

The activation function is applied to introduce nonlinearity:

$$A^{(l)} = f\left(Z^{(l)}\right) \tag{14}$$

where $f(.)$ represents the activation function.

Loss Function Calculation: The network is trained using categorical cross-entropy loss:

$$L = \sum_{i=1}^{N} \sum_{K=1}^{K} yi, ky_{i,k}\ log\ \hat{y}i_{,k} \tag{15}$$

where $y_{i,k}$ is the ground truth label for class k of sample i, N is the number of training samples, and K is the number of classes.

Backpropagation: To update the network parameters, we compute the gradient of the loss function concerning the weights using the chain rule:

$$\frac{\partial L}{\partial W^{(l)}} = \frac{\partial L}{\partial A^{(l)}} \times \frac{\partial A^{(l)}}{\partial Z^{(l)}} \times \frac{\partial Z^{(l)}}{\partial W^{(l)}} \tag{16}$$

Weight Update Using Gradient Descent: The network parameters are updated using the stochastic gradient descent (SGD) optimization method:

$$W^{(l)} \leftarrow W^{(l)} - \eta \frac{\partial L}{\partial W^{(l)}}$$
$$b^{(l)} \leftarrow b^{(l)} - \eta \frac{\partial L}{\partial b^{(l)}}$$

where η is the learning rate.

Iteration until Convergence: The training process iterates over multiple epochs until the loss function converges to a minimal value, ensuring the optimal performance of the model.

3.8. Performance Evaluation

The developed model is evaluated to measure how well it supports a solution to the problem. The prototype model is evaluated using the test set images from collected dataset images. The major components, such as the classification of disease, are evaluated. We used a confusion matrix to describe the classification efficiency of the model. Subsequently, we used precision, recall, and F1-score values to evaluate and discuss classification accuracy. Equations (3)–(6) present the formulas for calculating the mentioned metrics.

Accuracy is defined as the ratio of correctly recognized to total samples.

$$\frac{TP + TN}{TP + FP + FN + TN} \tag{17}$$

Classification error/misclassification is defined as the ratio of incorrectly recognized (either positive or negative) to total samples.

$$\frac{(FP + FN)}{(TP + TN + FP + FN)} \times 100 \tag{18}$$

Precision (or positive predictive value) is the proportion of predicted positives that are actually positive.

$$\frac{TP}{TP + FP} \tag{19}$$

Recall is the proportion of actual positives that are predicted as positive.

$$\frac{TP}{TP + FN} \tag{20}$$

We evaluated the improvement of preprocessing techniques using the following performance metrics.

Peak signal-to-noise ratio (PSNR) is used between the original and the reconstructed image as a quality indicator. The higher the PSNR, the better the quality of the compressed or reconstructed image [26].

Mean square error (MSE) is the average of the squared error that is used as the loss function for least-squares regression. There are no acceptable limits for MSE except that the lower the MSE, the higher the accuracy [26,27].

Structural similarity (SSIM) is a method used to measure the similarity between two images. The SSIM values range from 0 to 1, where 1 means a perfect match between the reconstructed image and the original one [26].

4. Model Evaluation and Discussion

In this section, we discuss the three different experiments performed to evaluate the model.

4.1. Performance of Proposed Model Trained with Original Images

As shown in the graph in Figure 6, both validation accuracy and training accuracy increase when we move from the first epoch to the last epoch. On the other hand, both validation loss and training loss decrease. In this experiment, the validation accuracy and training accuracy in the final batch are 92.22% and 94%, respectively. This indicates that the model can effectively learn from the training data and generalize well to the validation data, suggesting good convergence of the training process. Also, the validation loss and training loss in the final batch are 0.2319 and 0.1485, respectively. The consistent decrease in both training and validation loss further confirms the convergence of the optimization process. The close alignment between the training and validation metrics, with the validation performance closely tracking the training performance, suggests that the model is not overfitting the training data. In Figure 6, the black line at the top corresponds to the training accuracy, while the blue line indicates the validation accuracy. Below that, the red line represents the validation loss, and the black line shows the training loss.

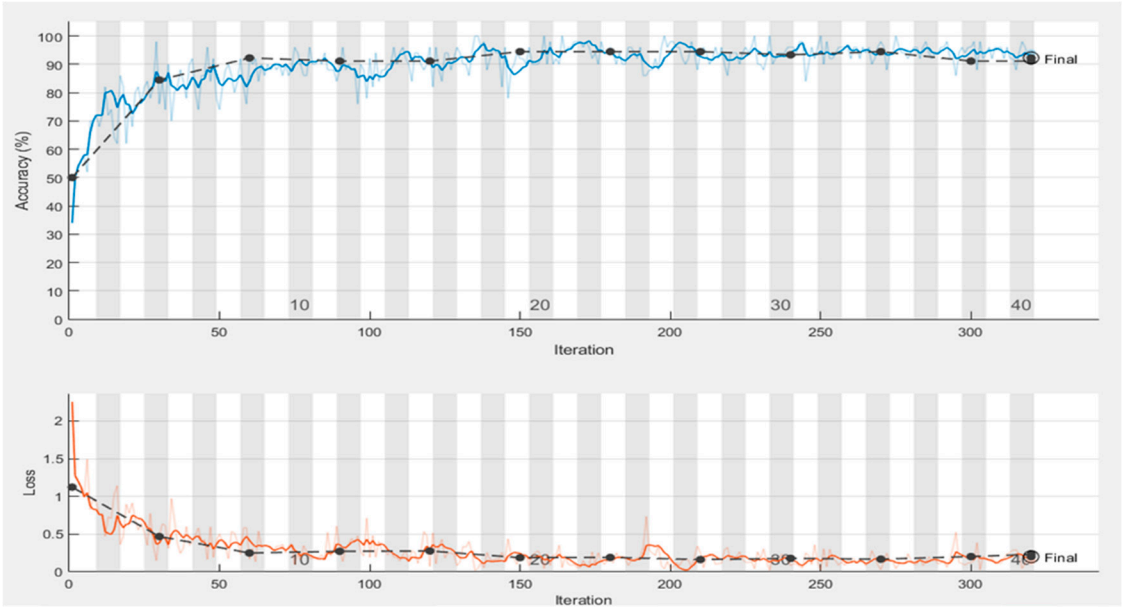


Figure 6. Training and validation accuracy using original images.

As shown in Table 1, the model was 90% accurate when tested with the test set data. This classification accuracy was achieved when the model was trained without preprocessing, segmentation, or data augmentation. The simulation took 24 min to train (36 s per each epoch (40)). In experiments 1 and 2, 360 testing images were employed and divided into three classes: phyllody disease class, bacterial blight disease class, and healthy class. In experiment 1, 36 images out of the 360 sesame leaf images were misclassified.

Table 1. Testing accuracy of original images.

Epoch	Iteration	Time Elapsed	Mini-Batch Accuracy	Validation Accuracy	Mini-Batch Loss	Validation Loss
1	1	0:00:43	34.00%	50.00%	2.2525	1.1228
4	30	0:03:12	76.00%	84.44%	0.4858	0.4692
7	50	0:04:46	74.00%	92.22%	0.5023	0.2504
8	60	0:05:35	92.00%	92.22%	0.1682	0.2504
12	90	0:08:02	88.00%	91.11%	0.3048	0.2727
13	100	0:08:49	92.00%	91.11%	0.1922	0.2776
15	120	0:10:20	96.00%	94.44%	0.0737	0.1889
19	150	0:12:36	92.00%	94.44%	0.2408	0.1923
23	180	0:14:38	96.00%	94.44%	0.1399	0.1923
25	200	0:15:58	98.00%	94.44%	0.042	0.1641
27	210	0:16:40	96.00%	93.33%	0.0835	0.1793
30	240	0:18:40	94.00%	94.44%	0.0284	0.1692
32	250	0:19:19	100.00%	94.44%	0.0299	0.1692
34	270	0:20:38	98.00%	91.11%	0.1247	0.2036
38	300	0:22:38	94.00%	91.11%	0.1485	0.2319
40	320	0:24:01	94.00%	91.11%	0.1485	0.2319
Test accuracy 0.9						

4.2. Performance of Proposed Model Trained with Preprocessed and Segmented Images

In experiment 2, the model was trained using the preprocessed and SegNet-based segmented images. The validation accuracy was 95.56%. It was increased by 3% compared to that of experiment 1. As Figure 7 shows, the training accuracy and testing accuracy were 96% and 94.44%, respectively. The training and testing accuracy improved by 2% and 4%. The validation loss and training loss in the final batch were 0.5162 and 0.1672, respectively. The simulation took 21 min and 34 s to train (32 s per each epoch (40)), which was much better than the above experiments. The graphs demonstrate promising results in terms of the model's ability to converge to high accuracy and low loss values on both the training and validation data. In Figure 7, the black line at the top corresponds to the validation accuracy, while the blue line indicates the training accuracy. Below that, the red line represents the training loss, and the black line shows the validation loss.

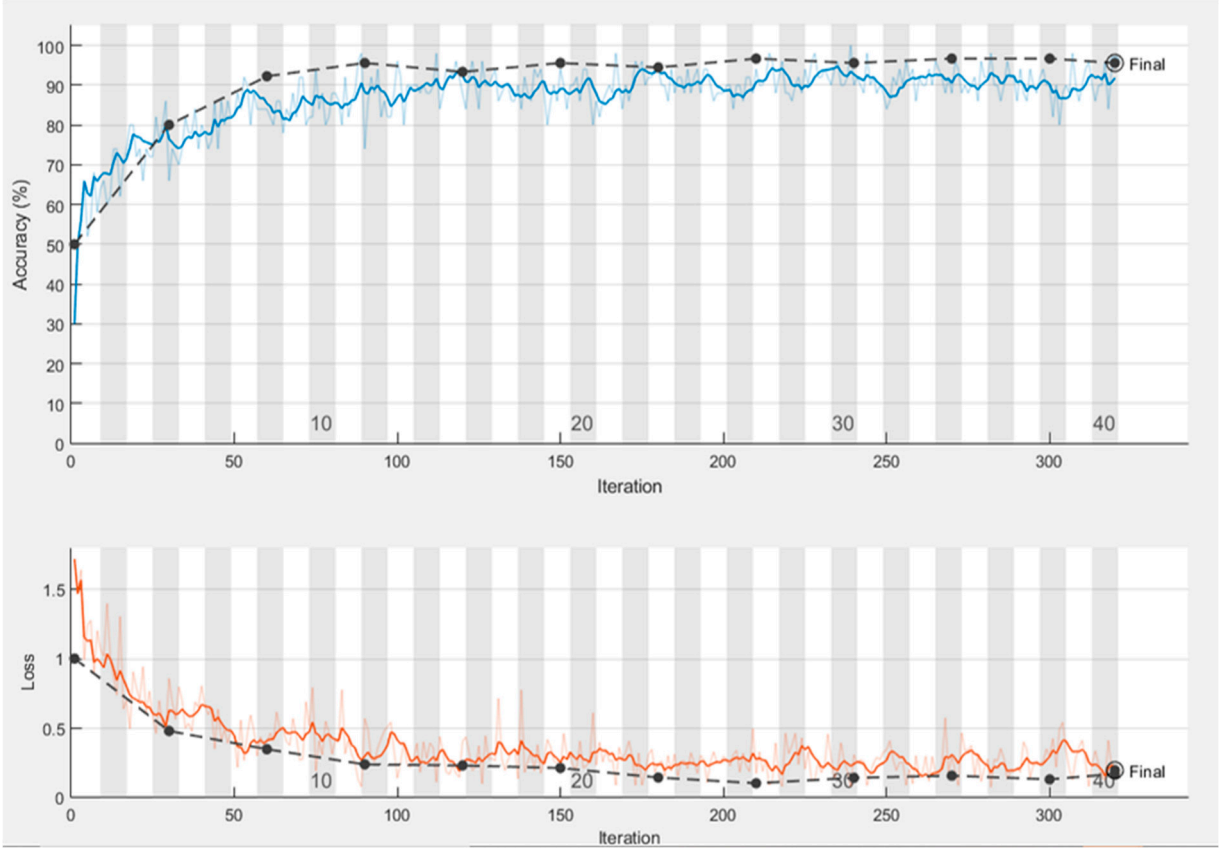


Figure 7. Training and validation accuracy using preprocessed and segmented images.

In Table 2, the graph shows the validation accuracy, validation loss, training accuracy, and training loss from the first epoch to the last epoch in the training progress when the model was trained with preprocessed and segmented images.

Table 2. Testing accuracy of preprocessed and segmented images.

Epoch	Iteration	Time Elapsed	Mini-Batch Accuracy	Validation Accuracy	Mini-Batch Loss	Validation Loss
1	1	0:00:48	28.00%	60.00%	1.7889	0.8766
4	30	0:03:12	78.00%	81.11%	0.5076	0.5075
7	50	0:04:36	88.00%	87.78%	0.2883	0.5753
8	60	0:05:20	86.00%	87.78%	0.394	0.5753

Table 2. *Cont.*

Epoch	Iteration	Time Elapsed	Mini-Batch Accuracy	Validation Accuracy	Mini-Batch Loss	Validation Loss
12	90	0:07:27	70.00%	90.00%	0.7554	0.4131
13	100	0:08:08	90.00%	91.11%	0.19	0.4551
15	120	0:09:30	98.00%	91.11%	0.0656	0.396
19	150	0:11:27	86.00%	92.22%	0.2381	0.4348
23	180	0:13:23	88.00%	88.89%	0.3693	0.5112
25	200	0:14:38	100.00%	88.89%	0.0447	0.5112
27	210	0:15:19	94.00%	91.11%	0.2933	0.4465
30	240	0:17:22	92.00%	91.11%	0.1948	0.4905
32	250	0:18:00	94.00%	90.00%	0.2031	0.4905
34	270	0:19:19	96.00%	91.11%	0.2284	0.4856
38	300	0:21:15	96.00%	91.11%	0.1672	0.5162
40	320	0:22:34	96.00%	91.11%	0.1672	0.5162
Test accuracy 0.9444						

4.3. Performance of Proposed Model Trained with Augmented Images

In experiment 3, as Figure 8 shows, the validation accuracy and training accuracy were 97.78% and 98% respectively. Also, the validation loss and training loss in the final batch were 0.0466 and 0.2090, respectively. The consistent decrease in both training and validation loss, with a final validation loss of 0.05 and training loss of 0.21, further confirms the optimization process. Compared to the previous experiment, the model has improved its validation accuracy by 2.22 percentage points and reduced the validation loss from 0.52 to 0.05, suggesting continued progress. To enable a more comprehensive analysis and comparison with existing methods, additional details on the neural network architecture, hyperparameters, and evaluation of other datasets would be helpful in assessing the strengths and limitations of the proposed approach and its potential for broader application. In Figure 8, the black line at the top corresponds to the validation accuracy, while the blue line indicates the training accuracy. Below that, the red line represents the training loss, and the black line shows the validation loss.

Also, Table 3 shows that the model was 96.67% accurate when tested with the test set data. In this experiment, 1440 testing images were employed and divided into three classes: phyllody disease class, bacterial blight disease class, and healthy class. Only 47 images out of the 1440 sesame leaf images were misclassified. The training and testing accuracy each improved by 2% as compared with that of the model trained before augmentation. This is because augmentation allows the model to train more features and memorize exact details of the images. The experimental results indicate that the model performance here was better for the recognition and classification of sesame disease. The training accuracy, validation accuracy, and testing accuracy were almost equal; therefore, there was no overfitting or underfitting problem in this experiment.

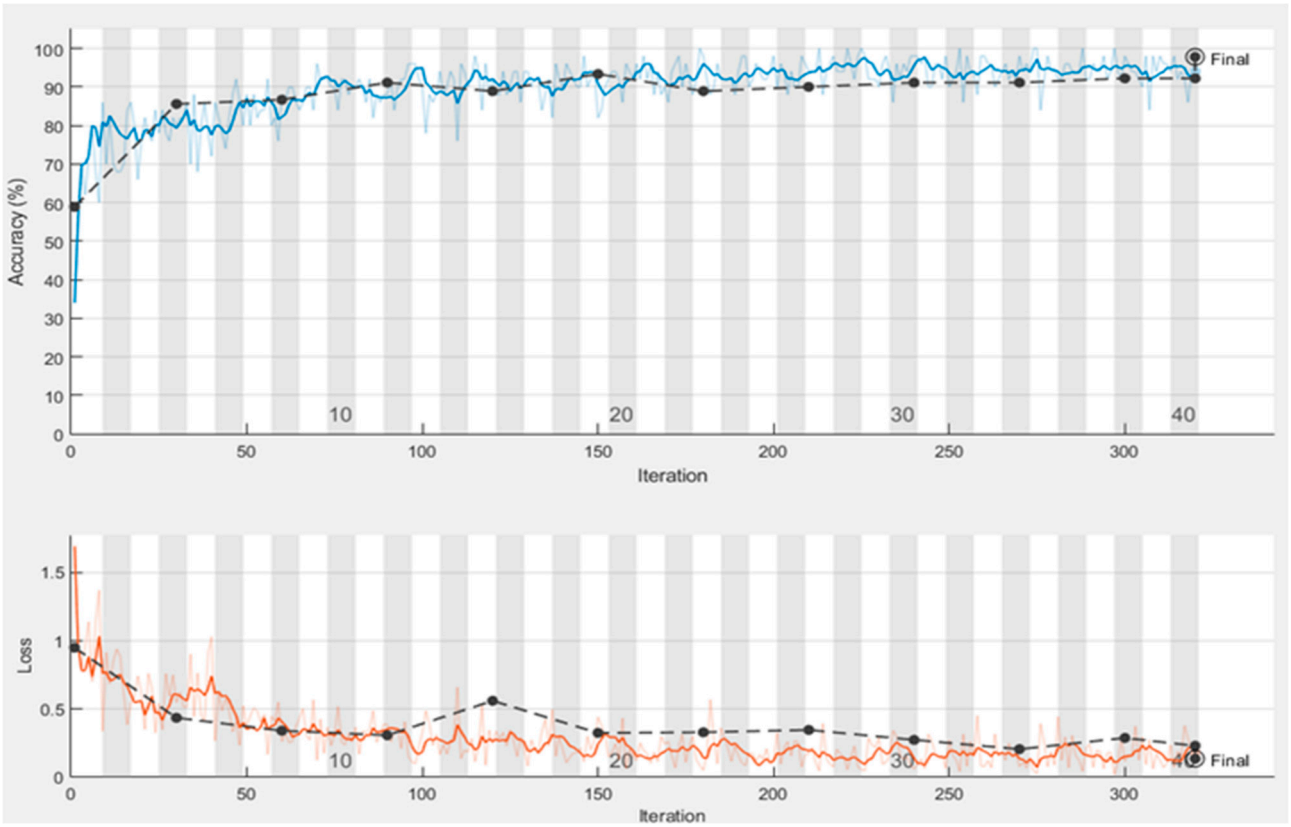


Figure 8. Training and validation accuracy using augmented images.

Table 3. Testing accuracy of augmented images.

Epoch	Iteration	Time Elapsed	Mini-Batch Accuracy	Validation Accuracy	Mini-Batch Loss	Validation Loss
1	1	0:00:28	38.00%	56.67%	1.9619	1.1099
4	30	0:02:53	86.00%	86.67%	0.3858	0.3753
7	50	0:04:10	74.00%	84.44%	0.7992	0.433
8	60	0:04:52	90.00%	86.67%	0.4536	0.3306
12	90	0:06:51	92.00%	86.67%	0.138	0.3306
13	100	0:07:29	74.00%	84.44%	1.0104	0.3789
15	120	0:08:47	90.00%	91.11%	0.2837	0.2539
19	150	0:10:47	74.00%	86.67%	0.1437	0.2942
23	180	0:12:39	92.00%	88.89%	0.2929	0.229
25	200	0:14:40	92.00%	90.00%	0.2897	0.2327
27	210	0:16:43	94.00%	90.00%	0.145	0.2327
30	240	0:16:43	96.00%	94.44%	0.1825	0.1961
32	250	0:18:13	92.00%	94.44%	0.1825	0.1961
34	270	0:18:49	92.00%	93.33%	0.1449	0.2185
38	300	0:21:13	98.00%	92.22%	0.0466	0.209
40	320	0:23:13	98.00%	92.22%	0.0466	0.209
Test accuracy 0.9667						

4.4. Discussion

4.4.1. Comparison with Inception-V3 Implementation Result

InceptionV3 is a depth-based convolution neural architecture with 24 million parameters. It uses small and asymmetric filters (1×7 and 1×5) and a 1×1 convolution as a bottleneck before the large filters. In Inception-V3, a 1×1 convolutional operation was used, which captured the cross-feature map and the spatial correlations within each channel via 3×3 or 5×5 regular convolutions. It is composed of 48 layers, and the network stacks 11 inception modules, where each module consists of pooling layers and convolutional filters with rectified linear units as an activation function. The input image of the model is a 2D sesame crop image with an input size of 299 by 299. In our case, we have only three classes, but the model can classify images up to 1000 categories. At the final concatenation layer, three fully connected layers of size 1024, 512, and 3 are added to prepare the output. As shown in Figure 9, the validation accuracy and training accuracy are 88.89% and 92%, respectively. The validation loss and training loss in the final batch are 0.2446 and 0.1778, respectively. In Figure 9, the black line at the top corresponds to the validation accuracy, while the blue line indicates the training accuracy. Below that, the red line represents the training loss, and the black line shows the validation loss.

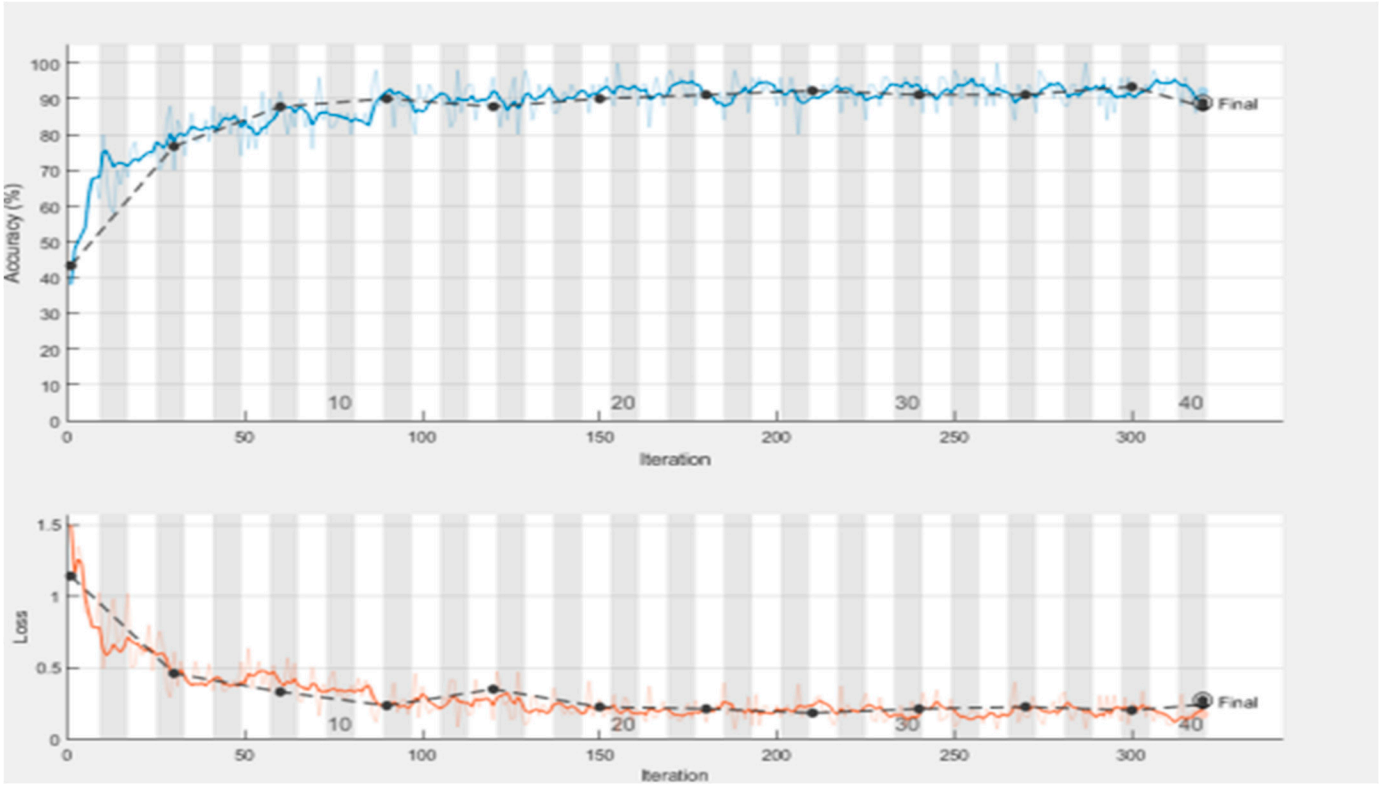


Figure 9. Inception-V3 model training and validation accuracy using augmented images.

Also, the test accuracy in Table 4 illustrates that the model was 88.89% accurate. When compared with the proposed model, the training and testing accuracy was decreased by 6% and 8%, respectively. This model took around 6 h and 33 min to train, with 589 s for each epoch (40). The accuracy is lower than that of the proposed model, which has a training and testing accuracy of 98% and 96.67%, respectively. It also takes a lot more time to compute.

Table 4. InceptionV3 model testing accuracy using augmented images.

Epoch	Iteration	Time Elapsed	Mini-Batch Accuracy	Validation Accuracy	Mini-Batch Loss	Validation Loss
1	1	00:00:38	38.00%	43.33%	1.4999	1.1412
4	30	00:02:46	70.00%	76.67%	0.6124	0.4623
7	50	00:04:09	88.00%	87.78%	0.387	0.3339
8	60	00:04:53	92.00%	87.78%	0.2985	0.3339
12	90	00:06:58	92.00%	90.00%	0.2654	0.2376
13	100	00:07:38	90.00%	87.78%	0.3719	0.3521
15	120	00:09:05	96.00%	90.00%	0.1376	0.2287
19	150	00:11:10	94.00%	91.11%	0.2833	0.2157
23	180	00:13:17	92.00%	92.22%	0.1218	0.1865
25	200	00:14:38	92.00%	88.89%	0.2385	0.229
27	210	00:15:28	96.00%	91.11%	0.2519	0.2152
30	240	00:17:28	96.00%	91.11%	0.1583	0.2295
32	250	00:18:06	92.00%	91.11%	0.0896	0.2295
34	270	00:19:25	92.00%	93.33%	0.1573	0.205
38	300	00:21:22	90.00%	87.78%	0.2623	0.2446
40	320	00:22:40	92.00%	87.78%	0.1778	0.2446
Test accuracy 0.8889						

4.4.2. Comparison with Xception Implementation Result

The Xception convolutional neural network can classify images into 1000 class categories. Here, the different spatial dimensions (1×1, 5×5, 3×3) of the Inception architecture are replaced with a single dimension (3×3) followed by a 1×1 convolution. In the 3×3 convolutions, spatial correlations within each channel are captured, and in the 1×1 convolutions, the cross-feature map is captured. This replacement architecture allows it to regulate its computational complexity. It has 71 layers with 23 million parameters. From those layers, the 36 convolutional layers form the feature extraction base of the network and have an input image size of 299 by 299 [27]. As shown in Figure 10, the validation accuracy and training accuracy are 92.2% and 94%, respectively. The validation loss and training loss in the final batch are 0.2316 and 0.1850, respectively. In Figure 10, the black line at the top corresponds to the validation accuracy, while the blue line indicates the training accuracy. Below that, the red line represents the training loss, and the black line shows the validation loss.

Also, Table 5 shows that the test accuracy of the model was 93.3%. When compared with the proposed model, the training and testing accuracy was decreased by 4% and 3%, respectively. This model took around 5 h and 57 min to train, with 535 s for each epoch (40). The accuracy is lower than the proposed model, and takes a lot more time to compute.

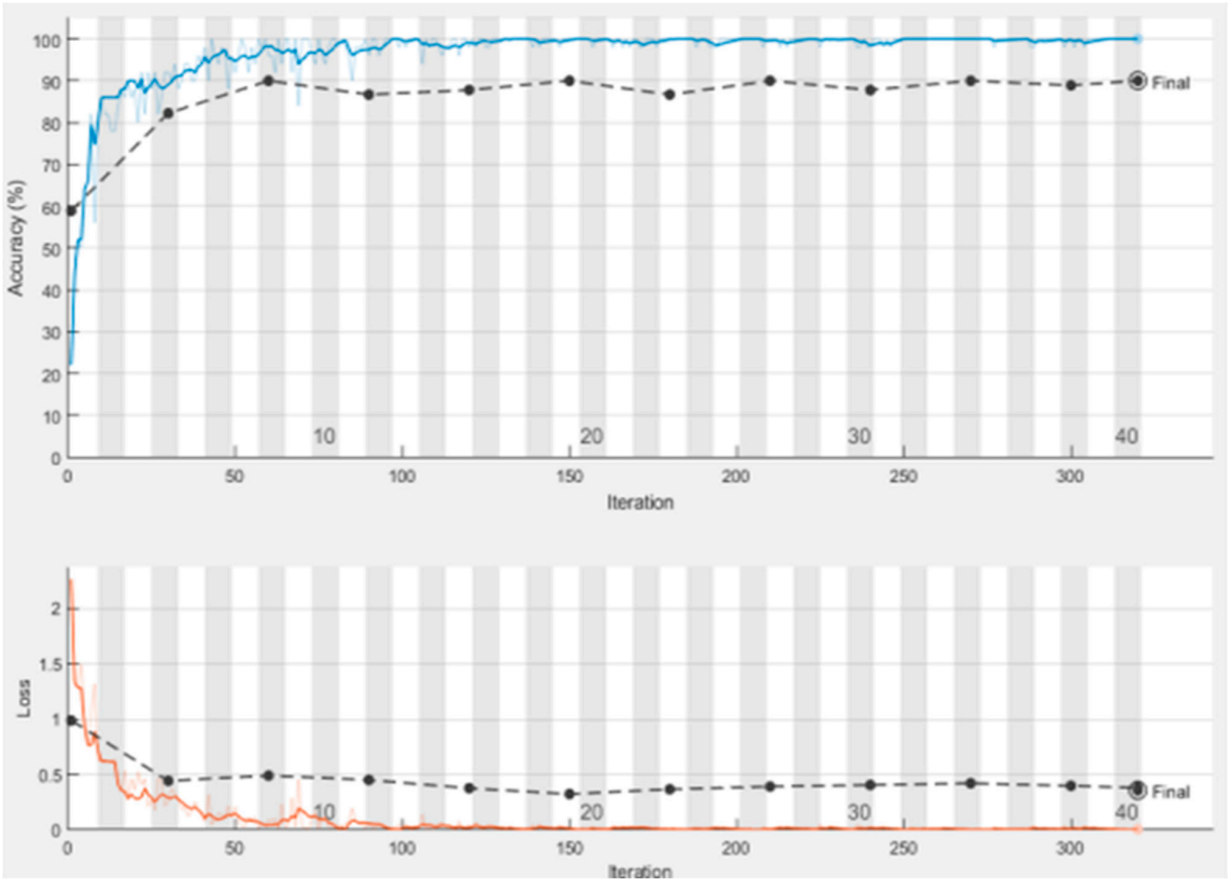


Figure 10. Xception model training and validation accuracy using augmented images.

Table 5. Xception model testing accuracy using augmented images.

Epoch	Iteration	Time Elapsed (hh:mm:ss)	Mini-Batch Accuracy	Validation Accuracy	Mini-Batch Loss	Validation Loss
1	1	00:00:22	34.00%	58.89%	1.7006	0.9504
4	30	00:02:19	80.00%	85.56%	0.5491	0.4368
7	50	00:04:00	88.00%	88.89%	0.367	0.3421
8	60	00:04:18	88.00%	86.67%	0.2642	0.3088
12	90	00:06:18	88.00%	91.11%	0.388	0.5609
13	100	00:07:06	90.00%	88.89%	0.2456	0.3295
15	120	00:08:15	92.00%	93.33%	0.4419	0.3259
19	150	00:10:14	94.00%	88.89%	0.0522	0.3477
23	180	00:12:13	92.00%	90.00%	0.1239	0.3477
25	200	00:13:41	94.00%	90.00%	0.0715	0.2749
27	210	00:14:12	94.00%	91.11%	0.1271	0.2749
30	240	00:16:48	90.00%	91.11%	0.2741	0.2749
32	250	00:17:18	90.00%	91.11%	0.2714	0.2749
35	270	00:19:10	98.00%	92.22%	0.118	0.2396
38	300	00:20:11	98.00%	92.22%	0.185	0.2316
40	320	00:21:34	94.00%	92.22%	0.185	0.2316
Test accuracy 0.9333						

As shown in Table 1, the image preprocessing and segmentation methods that were used in our experiment improved the performance of both classifiers. Also, the image augmentation process that was used to increase the number of images in the dataset improved the classification performance. In addition, the precision and recall results illustrate that the proposed model performs better in recalling the positive samples and also in detecting samples, as most of them were positive samples. This result illustrates that the proposed model achieved better accuracy than the two previous models as shown in Table 6. This is because, as we discussed in Section 3, the layers and parameters of the proposed model were selected based on the problem domain that we wanted to study. In addition, the proposed model has 1.1 million parameters, which is much smaller than the Xception model, which has 23 million parameters, and the InceptionV3 model, which has 24 million parameters. The smaller size of the proposed model makes it more efficient, especially in terms of computational resource usage such as memory. The misclassification in the proposed method is due to the presence of noise or artifacts in the images, which can confuse the model, leading to incorrect predictions. The improper tuning of hyperparameters during training can hinder the model's ability to generalize effectively, and changes in lighting, background, or other environmental factors during image capture can impact the model's performance on real-world data.

Table 6. Summary performance result of proposed model, Xception, and Inception-V3.

Trained with the Original Images						
Model	**Precision**	**Recall**	**F1-Score**	**Testing Accuracy**	**Training Accuracy**	**Validation Accuracy**
Proposed model	0.90	0.90	0.90	90.0%	94.0%	92.2%
InceptionV3	0.73	0.73	0.73	73.3%	82.0%	75.6%
Xception	0.89	0.89	0.89	88.9%	92.0%	86.7%
Trained with Preprocessed and Segmented Images						
Model	**Precision**	**Recall**	**F1-Score**	**Testing Accuracy**	**Training Accuracy**	**Validation Accuracy**
Proposed model	0.94	0.94	0.94	94.4%	96.0%	95.6%
InceptionV3	0.82	0.82	0.82	82.2%	92.2%	86.7%
Xception	0.90	0.90	0.90	90.0%	92.0%	90.0%
Trained with Augmented Images						
Model	**Precision**	**Recall**	**F1-Score**	**Testing Accuracy**	**Training Accuracy**	**Validation Accuracy**
Proposed model	0.97	0.97	0.97	96.7%	98.0%	97.7%
InceptionV3	0.89	0.89	0.89	88.9%	92.0%	88.9%
Xception	0.93	0.93	0.93	93.3%	94.0%	90.0%

4.4.3. Sample Classification of Sesame Plant Diseases

The Figure 11 shows the graphical user interface of the developed model, which shows an example of the classification of sesame plant diseases. The classification result is displayed at the bottom of the interface.

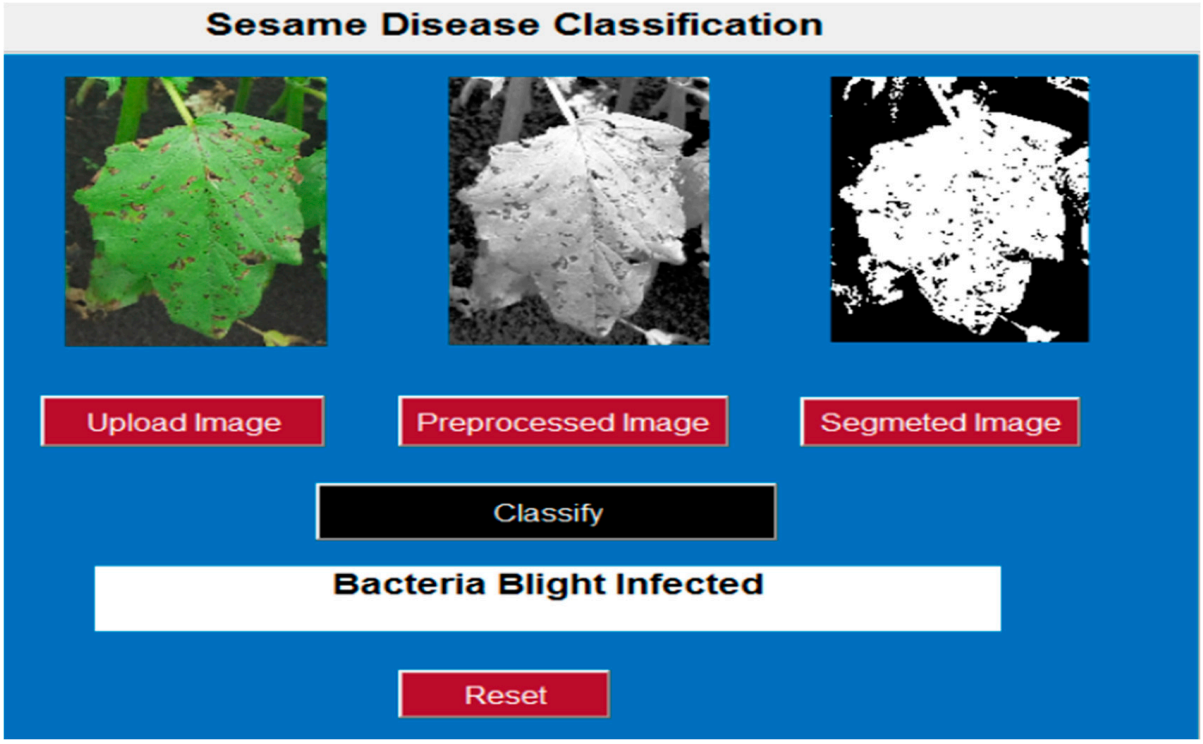


Figure 11. Sesame disease classification.

Figure 11 illustrates the categorization of sesame leaf disease through a system based on image processing and deep learning techniques. The process starts by uploading an image that displays a sesame leaf exhibiting clear disease symptoms. The image subsequently goes through preprocessing, which generally involves converting the image to grayscale, reducing noise, and enhancing contrast to emphasize features related to the disease. Subsequently, segmentation is carried out to distinguish the affected areas from the healthy sections of the leaf, facilitating the extraction of significant features for classification. Once these steps are completed, the system identifies the condition as "Bacteria Blight Infected", signifying that the sesame leaf is impacted by bacterial blight. The interface features buttons for uploading images, preprocessing, segmenting, classifying, and resetting the process, maintaining an organized workflow. The findings indicate that the system effectively recognizes and categorizes diseases, supporting early identification and possible management approaches.

5. Conclusions

Several kinds of sesame plant diseases are major reasons for the reduction in product quality. The early detection of diseases is important for taking action to prevent the spread of disease to other crops. In this study, we proposed a deep conventional neural network-based classification model for sesame leaf disease, i.e., bacterial blight and phyllody, using images. We evaluated the performance of the proposed model trained with the original images, trained with preprocessed and segmented images, and trained after image augmentation. The model achieved 90%, 94.44%, and 96.67% testing accuracy, respectively. The proposed model was compared with previous models, InceptionV3 and Xception. According to the results, the Xception model achieved 93.3% testing accuracy, and InceptionV3 achieved 90% when trained with the augmented images. Therefore, as one can see from the results, this study achieves good results for the classification of sesame leaf diseases.

Author Contributions: Conceptualization and methodology were led by E.A.N. and A.M.M., who also performed the state-of-the-art analysis. Software implementation was conducted by E.A.N., Resources, and formal analysis, data curation, and validation were carried out by K.K. and R.M.-B., Writing—original draft preparation was undertaken by E.A.N., A.M.M. and A.M.A., while writing—review & editing and administration—interoperability information and relevant use cases were managed by R.M.-B. and the A.M.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Ethical review and approval were waived for this study due to the nature of the data collection, which involved only plant images.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: The datasets generated and/or analyzed during the current study are not publicly available but are available from the corresponding author upon reasonable request.

Acknowledgments: This study forms part of the AGROALNEXT program and was supported by MCIN with funding from the European Union NextGenerationEU (PRTR-C17.I1) and by Fundación Séneca with funding from Comunidad Autónoma Región de Murcia (CARM).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Reyad, S.; Albakry, B.S.S. Sesame oil, properties and advantages (Review Article). *Futur. Biol.* **2022**, *1*, 1–9. [CrossRef]
2. Wei, P.; Zhao, F.; Wang, Z.; Wang, Q.; Chai, X.; Hou, G.; Meng, Q. Nutritional Value, Phytochemical Composition, Health Benefits, Development of Food, and Industrial Applications. *Nutr. Rev.* **2022**, *14*, 4079.
3. Caruelle, D.; Shams, P.; Gustafsson, A.; Lervik-Olsen, L. Affective Computing in Marketing: Practical Implications and Research Opportunities Afforded by Emotionally Intelligent Machines. *Mark. Lett.* **2022**, *33*, 163–169. [CrossRef]
4. Mequanenit, A.M.; Ayalew, A.M.; Salau, A.O.; Nibret, E.A.; Meshesha, M. Prediction of mung bean production using machine learning algorithms. *Heliyon* **2024**, *10*, e40971. [CrossRef]
5. Vinnichek, L.; Pogorelova, E.; Dergunov, A. Oilseed market: Global trends. *IOP Conf. Ser. Earth Environ. Sci.* **2019**, *274*, 012030. [CrossRef]
6. Meena, B.; Indiragandhi, P.; Ushakumari, R. Screening of sesame (*Sesamum indicum* L.) germplasm against major diseases. *J. Pharmacogn. Phytochem.* **2018**, *7*, 1466–1468.
7. Kindeya, Y.B.; Golla, W.N.; Kebede, A.A.; Sibhatu, F.B. Survey and Identification of Major Sesame Diseases in Low Land Areas of Western Zone of Tigray. *J. Biomater.* **2018**, *2*, 58–64. [CrossRef]
8. Ajay, A.; Reddy, K.V.S.R.; Kumar, A.A.; Someswar, C. Performance evaluation of twin-row planter for maize crop. *J. Eco-Friendly Agric.* **2024**, *19*, 214–218. [CrossRef]
9. Development of Automatic Sesame Grain Grading System Using Image | 37212. Available online: https://www.longdom.org/proceedings/development-of-automatic-sesame-grain-grading-system-using-image-processing-techniques-37212.html (accessed on 15 December 2024).
10. Alshawwa, I.A.; Elsharif, A.A.; Abu-Naser, S.S. An Expert System for Coconut Diseases Diagnosis. *Int. J. Acad. Eng. Res.* **2019**, *3*, 8–13.
11. Marcos, A.P.; Rodovalho, N.L.S.; Backes, A.R. Coffee Leaf Rust Detection Using Convolutional Neural Network. In Proceedings of the 2019 XV Workshop de Visão Computacional (WVC), São Bernardo do Campo, Brazil, 9–11 September 2019; pp. 38–42. [CrossRef]
12. Arivazhagan, S.; Ligi, S.V. Mango Leaf Diseases Identification Using Convolutional Neural Network. *Int. J. Pure Appl. Math.* **2018**, *120*, 11067–11079.
13. Zhang, X.; Qiao, Y.; Meng, F.; Fan, C.; Zhang, M. Identification of maize leaf diseases using improved deep convolutional neural networks. *IEEE Access* **2018**, *6*, 30370–30377. [CrossRef]
14. Yogananda, G.S.; Babu, A.J.; Julma, S.A.; Madhukar, G.; Manjula, B.M. Rice Plant Disease Classification using Transfer Learning of Deep ResNext Model. In Proceedings of the 2023 International Conference on Evolutionary Algorithms and Soft Computing Techniques (EASCT), Bengaluru, India, 20–21 October 2023; Volume XLII, pp. 18–20. [CrossRef]
15. Afonso, M.; Blok, P.M.; Polder, G.; Van Der Wolf, J.M.; Kamp, J. Blackleg Detection in Potato Plants using Convolutional Neural Networks. *IFAC-PapersOnLine* **2019**, *52*, 6–11. [CrossRef]

16. Idress, K.A.D.; Gadalla, O.A.A.; Öztekin, Y.B.; Baitu, G.P. Machine Learning-based for Automatic Detection of Corn-Plant Diseases Using Image Processing. *Tarim Bilim. Derg.* **2024**, *30*, 464–476. [CrossRef]
17. Munteanu, C.; Lazarescu, V. Evolutionary contrast stretching and detail enhancement of satellite images. *Proc. Mendel* **1999**, *99*, 94–99.
18. Sukassini, M.; Velmurugan, T. Noise removal using morphology and median filter methods in mammogram images. In Proceedings of the 3rd International Conference on Small and Medium Business, Hochiminh, Vietnam, 19–21 January 2016; pp. 413–419.
19. Arce, G.R.; Paredes, J.L. Recursive weighted median filters admitting negative weights and their optimization. *IEEE Trans. Signal Process.* **2000**, *48*, 768–779. [CrossRef]
20. Badrinarayanan, V.; Kendall, A.; Cipolla, R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 2481–2495. [CrossRef] [PubMed]
21. Guo, Y.; Liu, Y.; Georgiou, T.; Lew, M.S. A review of semantic segmentation using deep neural networks. *Int. J. Multimed. Inf. Retr.* **2018**, *7*, 87–93. [CrossRef]
22. Shorten, C.; Khoshgoftaar, T.M. A survey on Image Data Augmentation for Deep Learning. *J. Big Data* **2019**, *6*, 60. [CrossRef]
23. Toda, Y.; Okura, F. How convolutional neural networks diagnose plant disease. *Plant Phenomics* **2019**, *2019*, 9237136. [CrossRef] [PubMed]
24. Khan, A.; Sohail, A.; Zahoora, U.; Qureshi, A.S. A survey of the recent architectures of deep convolutional neural networks. *Artif. Intell. Rev.* **2020**, *53*, 5455–5516. [CrossRef]
25. Biswas, K.; Kumar, S.; Banerjee, S.; Pandey, A.K. Smooth Maximum Unit: Smooth Activation Function for Deep Networks using Smoothing Maximum Technique. In Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 784–793. [CrossRef]
26. Lai, L.; Liu, J. Support Vector Machine and Least Square Support Vector Machine Stock Forecasting Models. *Comput. Sci. Inf. Technol.* **2014**, *2*, 30–39. [CrossRef]
27. Ahn, J.M.; Kim, S.; Ahn, K.-S.; Cho, S.-H.; Lee, K.B.; Kim, U.S. Correction: A deep learning model for the detection of both advanced and early glaucoma using fundus photography. *PLoS ONE* **2019**, *14*, e0211579. [CrossRef] [PubMed]