Neural Computing and Applications (2023) 35:12055–12071
https://doi.org/10.1007/s00521-023-08339-w

ORIGINAL ARTICLE

Coconut trees classification based on height, inclination, and orientation using MIN-SVM algorithm

Rajesh Kannan Megalingam[1] · Sakthiprasad Kuttankulangara Manoharan[1] · Dasari Hema Teja Anirudh Babu[1] · Ghali Sriram[1] · Karanam Lokesh[1] · Sankardas Kariparambil Sudheesh[1]

Received: 15 July 2022 / Accepted: 25 January 2023 / Published online: 23 February 2023

Abstract

A computerized coconut tree detection system can help dendrologists and laypersons in identifying coconut trees based on three morphological parameters including height, inclination, and orientation. These three parameters help to determine the health and the nature of growth of coconut trees which influences the design and use of robots for harvesting coconuts. Deep learning is a powerful tool used for feature extraction as it is better in extracting deeper details (features) in an image. In this research work, a new Modified Inception Net based Hyper Tuning Support Vector Machine classification method named MIN-SVM is proposed for coconut tree classification based on three morphological parameters including height, inclination and orientation. The features from the pre-processed coconut tree images were extracted using four distinct Convolutional Neural Network models including Visual Geometry Group, Inception Net, ResNet, and MIN-SVM. These extracted features were then classified using a Machine Learning model named Support Vector Machine (SVM). The MIN-SVM have achieved a remarkable accuracy of 95.35 percent as contrasted to Visual Geometry Group (91.90%), Inception Net (81.66%), and ResNet (71.95%). The features extracted from Modified Inception Net fitted good with SVM classifier. Experimental results show that MIN-SVM can be powerful computerized automated system to identify coconut trees based on height, inclination, and orientation.

Keywords CNN · MIN-SVM · Inception net · ResNet · Feature extraction · Classification

1 Introduction

For years, machine learning is being used widely in agricultural fields in improving harvest quality, detecting and targeting weeds, minimizing losses, etc. Precision agriculture is used to analyze efficiency, quality, and productivity in farming and provide the decision support system to the entire farm management. In agricultural applications, machine learning algorithms are used for the classification of different types of fruits, plants, vegetables, etc. Until now there is very little research on coconut tree classification. Coconut palms are grown in more than 12.5 million hectares all over the world in more than 85 countries. The production is around 67.7 million nuts, and the productivity is more than 500 nuts per hectare. Farm productivity can increase the competitiveness of the coconut and also the income of the coconut farmers. Improved varieties of the coconut palm that have capacity to result in high productivity, resistance to infectious diseases and adaptation to

✉ Rajesh Kannan Megalingam
rajeshm@am.amrita.edu

Sakthiprasad Kuttankulangara Manoharan
Sakthiprasadkm@am.amrita.edu

Dasari Hema Teja Anirudh Babu
dasari.anirudh278@ieee.org

Ghali Sriram
sriramghali39@ieee.org

Karanam Lokesh
karanamlokesh@am.students.amrita.edu

Sankardas Kariparambil Sudheesh
kssankardas@am.students.amrita.edu

[1] Department of Electronics and Communication Engineering, AmritaVishwa Vidyapeetham, Amritapuri, India

climatic conditions and growing environments can help with higher productivity. Climatic conditions like wind, sunlight and rain can affect the height, inclination and orientation of the coconut trees.

In this research work, we are classifying coconut trees based on three morphological parameters which are height, inclination, and orientation using the proposed Modified Inception Net based Hyper Tuning Support Vector Machine (MIN-SVM) classification model. Height, inclination and orientation are very important parameters to be considered while designing robotic harvesters to harvest coconuts. This approach of coconut tree classification can help dendrologists and laypersons in identifying coconut trees along with its productivity. Pattern based image classification method requires huge dataset. The chance of overlapping of samples among the classes can be too high which affect accuracy and robustness. Due to the size of the dataset, computation complexity can be high in terms of time and space requirements. Currently there is no public dataset available for coconut trees. Creating a dataset for coconut trees can lead to comprehensive details of coconut palm research and development. As part of this research, we created a dataset containing seventeen thousand images of coconut trees, which can be used for future coconut tree-related research work.

Machine learning algorithms are widely used in agriculture fields to classify crops, plants and trees, and also identify diseases. They are used frequently to classify data trained on models like Support Vector Machine (SVM), K-nearest neighbors (K-NN), Random Forest, Naive Bayes, etc., to make predictions for future outcomes. Deep learning is one of the best neural networks to handle datasets with more samples which sometimes achieves better results than human-level performance. In deep learning, the models are trained using labeled data and neural networks that contain many layers. Using deep learning we can eliminate manual feature extraction resulting in the features being extracted while training the data. In the paper [1], the authors modified the Alex Net and proposed a new CNN model for both feature extraction and classification of Mango leaves infected by anthracnose disease. In our method Modified Inception Net is used for feature extraction and SVM is used for classification. Modified inception net is giving better features for classification compared to other deep learning models.

In this research work, hyper tuning Support Vector Machine (SVM) with Radial Basis Function (RBF) as kernel along with three coconut tree parameters including height, inclination, and orientation are used to classify the coconut trees. To train SVM, the GridSearchCV model is used. Using C and gamma hyper parameter combinations obtained from GridSearchCV, several probability distribution functions like Dweibull probability distribution, exponential probability distribution, normal probability distribution, Pareto probability distribution, and gamma probability distribution were calculated to find the relation between accuracy, C, and gamma combinations. Inception net is a pre-trained model in Convolutional Neural Network (CNN) which is composed of multiple layers. As part of this research work, three layers of feature extraction models, namely Local Binary Pattern (LBP), Histogram of Oriented Gradient (HOG), and Principal Component Analysis (PCA), were added prior to Inception net V3 to increase the performance of the model. The extracted deep features were fed into the hyper tuning SVM for classification. The Modified Inception Net based Hyper tuning Support Vector Machine (MIN-SVM) acquired best results compared to other algorithms.

2 Background study

Windrimet al. [2] developed a machine learning method for classifying health, detecting individual trees, and detecting dead trees. From SVM, KNN, naïve bayes, and GBDT (Gradient Boosting Decision Tree) algorithms, SVM gave the highest accuracy. In this method, the data preprocessing is byzantine due to the size of the large sensor data. Zhu et al. [3], developed a deep learning algorithm, ResNet using CNN (Convolutional Neural Network) and utilized the historical data of crop distribution map. The SVM is used for feature extraction. This proposed method is not evaluated in cases with no historical data. The paper [4] focused on measuring and estimating the diameter of a coconut tree trunk based on computer vision. The OpenCV methods, canny edge, erosion, dilution, Gaussian filter, and contours are used in this work. The proposed method with 83.63% accuracy is used for measurements and not for classifying the coconut trees. In one of our earlier research works [5], we presented a novel robotic coconut tree climber and harvester called Amaran. Amaran had accomplished climbing trees up to 15.2 m tall and tree trunk inclinations of up to 30°. But the coconut tree morphology is not considered during the design process of the robot. The robot's constraints are obtained from the structure's mechanical limitations. Fernández et al. [6], mainly focused on designing a robust algorithm to detect field-grown cucumbers for robotic harvesting. The proposed machine learning algorithm is based on several preprocessing and data mining techniques to get good accuracy. However even after using six preprocessing methods, the proposed algorithm achieved only 91.17% accuracy. The research paper [7] proposes a combination of deep learning and machine learning algorithms to classify tobacco crops for spraying pesticides. The feature extraction using CNN and classification using SVM give better results for their application. But the SVM parameters are not optimized in this proposed work. A novel deep learning-based method D-Leaf is introduced in [8]. Three types of CNN models, pre-trained Alex Net, fine-tuned Alex Net, and D-Leaf, are used for feature extraction and pre-processing. Compared to other algorithms, the D-leaf combined with the ANN model got better performance. Both the feature extraction and classification are performed by deep learning algorithms making this a computationally complex method.

Current state-of-the-art research methods use machine learning algorithms for classification in the field of agriculture. SVM is used for classification in many agricultural applications to identify medicinal plants using leaf features [9], prediction of crop yield [10], plant diseases detection [11], determining crop types and mapping [12], paddy rice detection and classification [13], leaf disease identification [14, 15]. Other than SVM, machine learning algorithms like Decision Trees to classify vineyards and agricultural objects [16], Random Forests for the identification of growth stages of cotton and chili [17], and KNN for wheat grain identification [18], Enhanced- KNN for classification of corn leaves to detect the disease [19], are used. Deep learning methods are also used in the field of agriculture like Inception Net with residual connections to detect plant diseases [20], EfficientNetV2 model for the identification of disease in cardamom plant leaves and grapes [21], CNN and Transfer learning with Alex Net to detect leaf disease [22], Hybrid Deep CNN transfer learning algorithm for classification of various rice diseases [23], CNN for pollen grains classification [24], CNN with the hierarchical transformation of the deep features for plant identification [25], Mask R-CNN to detect the leaf count and plant-specific leaf to estimate the plant growth stage [26], Convnets for plant disease detection [27], CNN based Inception V3 for the classification of corn leaf diseases [28], and PCA-Net to solve the problem of lack of an image database in the field of agriculture [29].

3 Methodology

Classifying coconut trees based on height, inclination and orientation morphological parameters using MIN-SVM algorithm consists of several steps.These steps include: pre-processing of the input image data, extracting deep features using LBP, HOG and PCA feature extractors and feeding to Inception Net, creation of training set and testing set, and training using hyper tuning SVM model. The overall system architecture is shown in Fig. 1. The first step is inputting the image data, which is collected from several trees. In the second step, pre-processing methods were carried out. In the background removal task, all the external noises which are dominating the coconut trees are removed using "rembg" and "grabcut" python packages. Resizing is an important task that should be carried out before inputting images into any algorithm. Images with an optimal size of (299 $\times$ 299) have been inputted into the algorithm. The third step is about extracting deep features of input data; LBP, HOG, and PCA feature extractors were used. These extracted features were fed into Inception Net to extract deep features. In the fourth step, the extracted features are split into a training set and testing set, with 70:30 ratios. The training set is sent into the hyper tuning SVM model and trained for classification in final step. We used Pareto probability distribution to find best C and gamma values for our data. Later the testing set is fed into the trained model (Hyper Tuning SVM). In the sixth step, the trained model gives classification of coconut trees. To classify the coconut trees using MIN-SVM algorithm, the dataset is divided into three different classes based on indexing.

For coconut trees with less than 5 m height the indexing value is 0.06. Indexing value of coconut trees with 5 m to 15 m height is 0.8. The indexing value is 0.14 for coconut trees with height more than 15 m. For coconut trees with inclination 0° to 10°, the indexing value is 0.6. For 10° to 30° inclination, the indexing value is 0.36. 0.04 is the indexing value for more than 30° inclination in coconut trees. If there is orientation in trees, the indexing value is 0.08. 0.92 is the indexing value if there is no orientation. The above shown (Tables 1, 2, 3) are indexing values of height, inclination, orientation. After acquiring the indexing values of height, inclination, and orientation, the final indexing value of the coconut trees is found by adding indexing values of height, inclination, and orientation. After obtaining the final indexing values of all trees, they are divided into three different classes. Range of class A is from 0.18 to 0.91. Class B ranges from 0.92 to 1.64, and range of class C is from 1.65 to 2.32.

3.1 Dataset collection and creation

To test and train our proposed MIN-SVM model, 17,000 images with several variations in height, inclination, and orientation are created using three thousand raw images (without augmentation) collected from one hundred and

Table 1 Indexing values of height

Height (meters)	Indexing value
Less than 5 m	0.06
5–15 m	0.8
Greater than 15 m	0.14

Table 2 Indexing values of inclination

Inclination (degrees)	Indexing value
0–10	0.6
10–30	0.36
Greater than 30	0.04

Table 3 Indexing values of orientation

Orientation	Indexing value
Yes	0.08
No	0.92

forty-two coconut trees. iPhone xR device with an equivalent focal length of 1.8 f, and a resolution of 12 MP is used to acquire coconut tree images. All the images are captured between 4:00 and 6:00 pm. Some of the images of the coconut trees are captured on beachland and some are captured on farmlands. The height of the trees varied between 3.5 to 15 m and inclination varied between 90 to 120 deg. All the images were collected between January and March 2022 in Vallikavu village, Kollam district, Kerala state of India.

3.1.1 Data collection of height of coconut trees

Figure 2 shows the approach to calculate the height manually. The height of each tree is measured with a constant distance of 2.4 m from the tree as shown in Fig. 2. Here h1 is 1.8 m which is the height of the person. The same person collected the data from all the trees. h2 is the distance from tree to the person which is 2.4 m. A constant distance h2 is used to calculate height for all the trees. Total height of the tree is sum of h1 and h3. To find h3, we calculated the inclination (θ) of person's neck when he sees the treetop. With inclination (θ) and h2, we found h3 using trigonometric formulas.

3.1.2 Data collection of inclination and orientation of coconut trees

To measure the inclination of the tree, an inclinometer is used for all the trees. Inclination is measured at 1.4 m from bottom of the coconut tree. An inbuilt Android app named 'Measure' is used to cross-check the obtained results.'Measure' application gave readings 95% similar to manually calculated readings. Orientation has been measured manually; if there is some twist or turn in the tree from ground to top of the tree, then the tree has orientation.

3.2 Image pre-processing

As part of our research work, three thousand raw images (without augmentation) collected from one hundred and forty-two coconut trees were pre-processed, which resulted in a total of 17,000 images. From one tree we took 22 images from different orientations, thus, a total of 3124 images from the 142 trees. The original images were pre-processed using three distinct methods, namely background removal, augmentation, and resizing. Figure 3 shows the steps involved in image pre-processing. The initial step holds data acquisition. In the second step, background removal method is performed to remove all the unnecessary plants, trees, and grass from the image. In the next step, two-position and three-color augmentation techniques were used. Translation and flipping methods were used under position augmentation techniques. Brightness, contrast, and sharpness methods were utilized under color augmentation techniques.

In translation the object in an image is shifted or moved to several directions up, down, left, and right. Figure 3 shows various image pre-processing techniques applied. Flipping augmentation involves mirror flipping objects in an image in left and right directions are also shown in Fig. 3. These techniques helped in creating a dataset of seventeen thousand images for testing and training. In the final step, all images were resized into (299 × 299 × 3) shape, which is an optimal size of an image to perform feature extraction using the proposed MIN-SVM model.

The creation of data set is to accommodate all the variations in the coconut trees in terms height, inclination, and orientation. The data set before application of augmentation methods consists of 3124 images. As increasing the dataset improves the learning rate in any deep learning method, we applied various augmentation methods like flipping, sharpness, brightness, etc. to make the proposed method more robust toward the variations in input. These pre-processing methods are required only during the training time. While testing, the raw images are the inputs for the proposed method, and all the test results mentioned in this paper used raw images.

We are measuring the parameters including height and inclination and checking the presence of the orientation of each tree. All 22 images of each tree and the labeling for each image is purely based on these parameters. The index value determines the classes and is composed of the index of height, inclination, and orientation. This indexing is based on the variation of coconut trees from a well-maintained farm. From the data of hundred trees, the normalized value of the number of trees belonging to each category is given as the index number. The images of these coconut trees (100 numbers) are not included in the dataset.

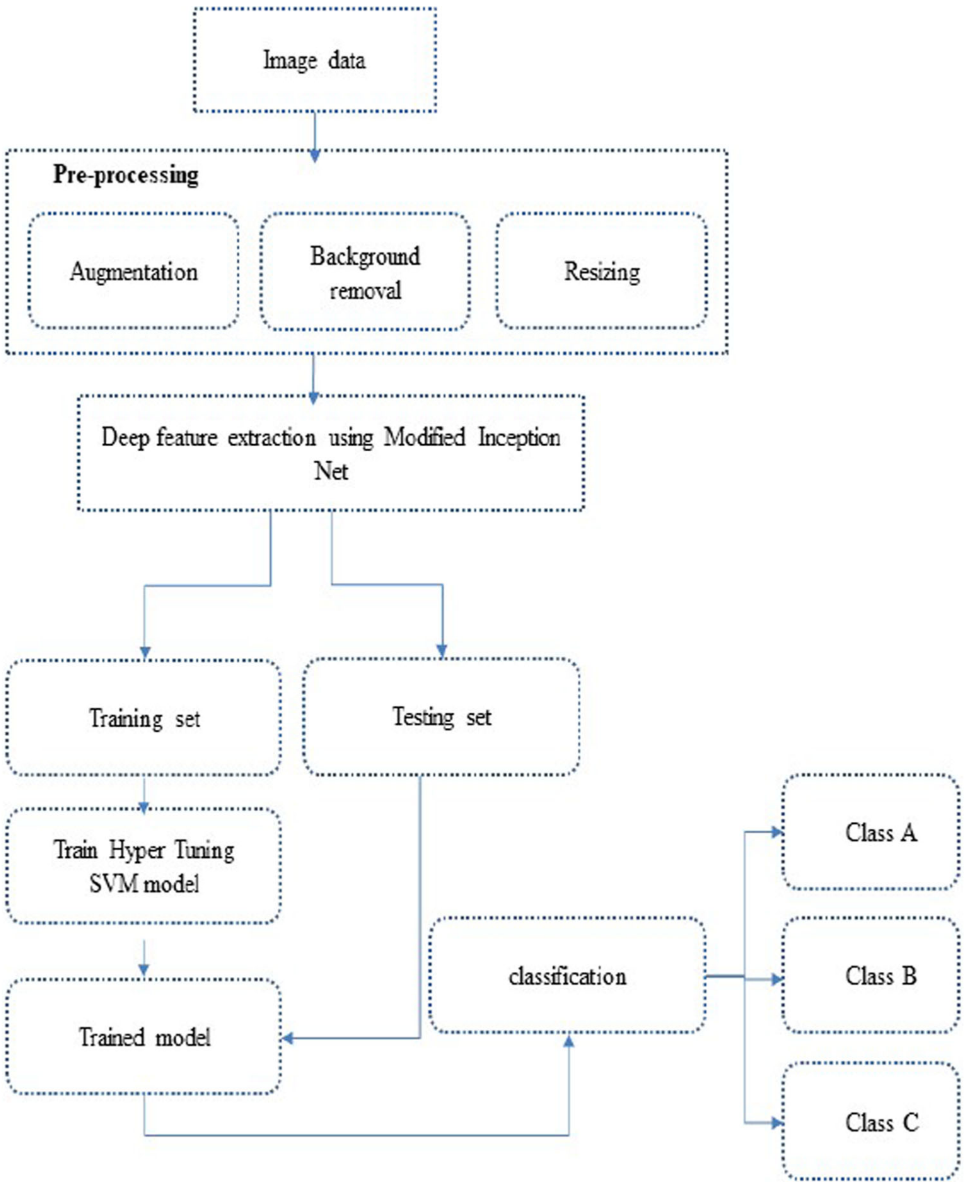

Fig. 1 System architecture of modified inception net based hyper tuning support vector machine (MIN-SVM) classification

3.3 Framework of traditional inception net

Figure 4 shows the traditional inception net architecture. In this case the pre-processed images are given directly to the inception net model. The pre-processed images are passed through all the 48 layers inside inception model. The output of inception net is a feature array of 2048 values. The output feature array along with target class is given to SVM for classification. In the case of the proposed MIN-SVM we added three more layers as shown in Fig. 5. Experiments and results show that Modified Inception net based Hyper Tuning Support Vector Machine gives more accuracy compared to traditional inception net.

3.4 Framework of proposed model (MIN-SVM)

Figure 5 shows the system architecture of “Modified Inception net V3”. In the first step, the pre-processed images are given as input to grayscale conversion and RGB conversion. Before going into inception net V3 the pre-processed images are passed through three more layers for feature extraction namely HOG, LBP and PCA. The first layer is LBP which is mainly used for texture classification. LBP labels the pixels of an image by a threshold value of the neighborhood of each pixel. When the LBP is combined with the HOG the classification performance improves significantly because, HOG gives orientation of each pixel along with gradient, thereby extracting unique features. The HOG is used as a feature descriptor for object detection, and it extracts the features from image data.

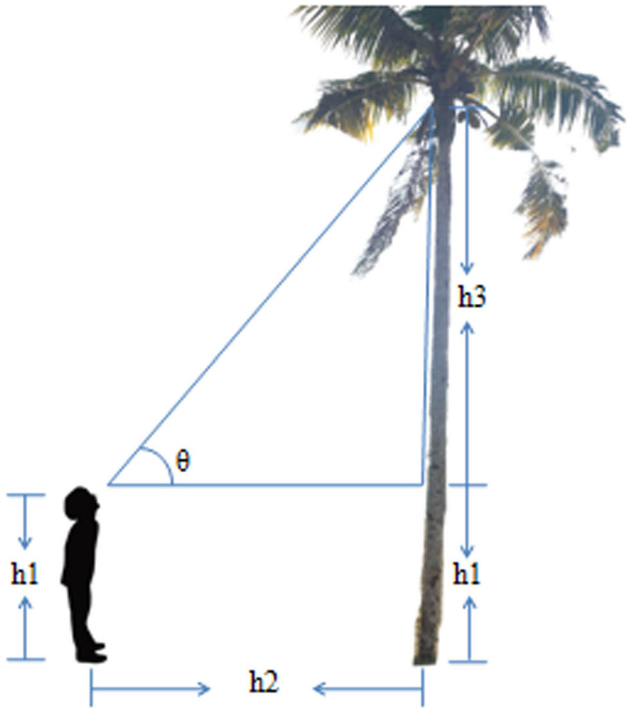

Fig. 2 Approach followed to calculate height of coconut trees

HOG breaks the whole image into smaller regions and calculates the gradient and orientation of each region. After calculating the gradient and orientation of all the regions, it creates a histogram for every region using gradient and orientation pixel values that are obtained. PCA which is the last feature extraction that we used to reduce the dimensionality of the feature array without losing any details of the feature array. Before giving the feature array to PCA, feature array must be standardized to make every value to be on the same scale. The three features that are extracted from images are appended to each other. The appended features are given as input to the Inception Net model to extract deep features. A deep feature is a node or layer's recurrent response to an input pertinent to the model's final output inside a hierarchical model. Depending on how early in the model or other structure the reaction is activated, one characteristic is seen as "deeper" than another. These features will have an analogy to the tree properties as these deep features are the output generated in response to the input tree features initially generated using the tree dataset. Inception net generates 2048 deep features file and corresponding target class file as output. Both deep features and target classes are given as input to the SVM for classification.

4 Mathematical modeling

4.1 Mathematical modeling of principal component analysis (PCA)

Pre-processed image is taken as input. All the features extracted from an image are appended to a common variable. The common variable 'F' holds all the features that are appended from three distinct feature extractions. C is the original input image, which is also known as red, blue, green (RBG) image. Here RBG values of each grid are

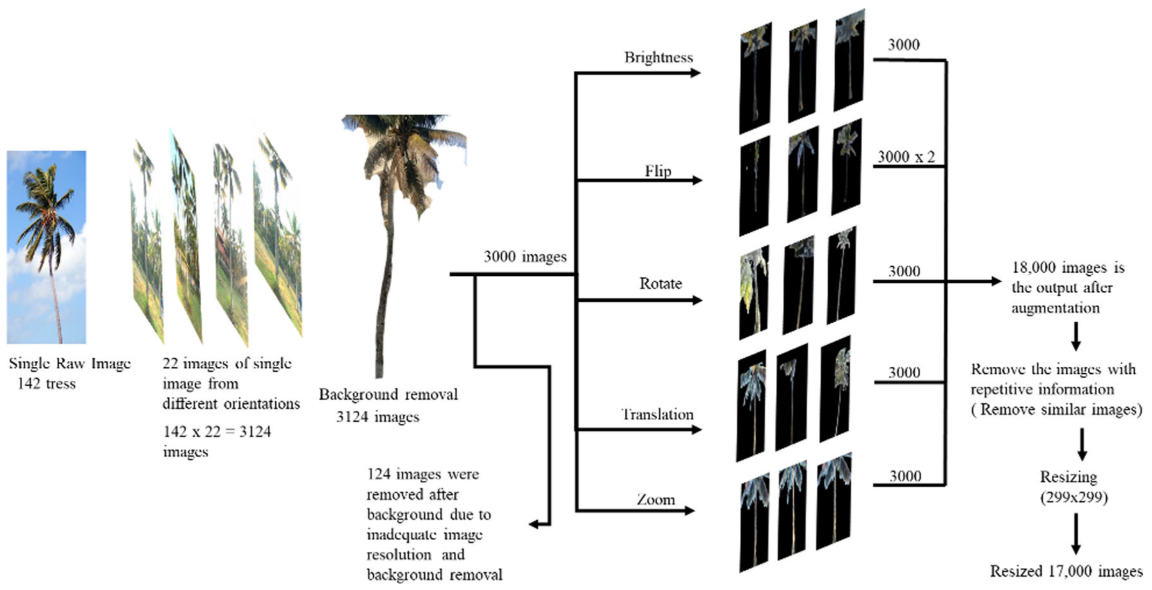
Single Raw Image
142 tress
22 images of single image from different orientations
142 x 22 = 3124 images
Background removal
3124 images
124 images were removed after background due to inadequate image resolution and background removal
3000 images
Brightness
Flip
Rotate
Translation
Zoom
3000
3000 x 2
3000
3000
3000
18,000 images is the output after augmentation
Remove the images with repetitive information (Remove similar images)
Resizing (299x299)
Resized 17,000 images

Fig. 3 Image pre-processing

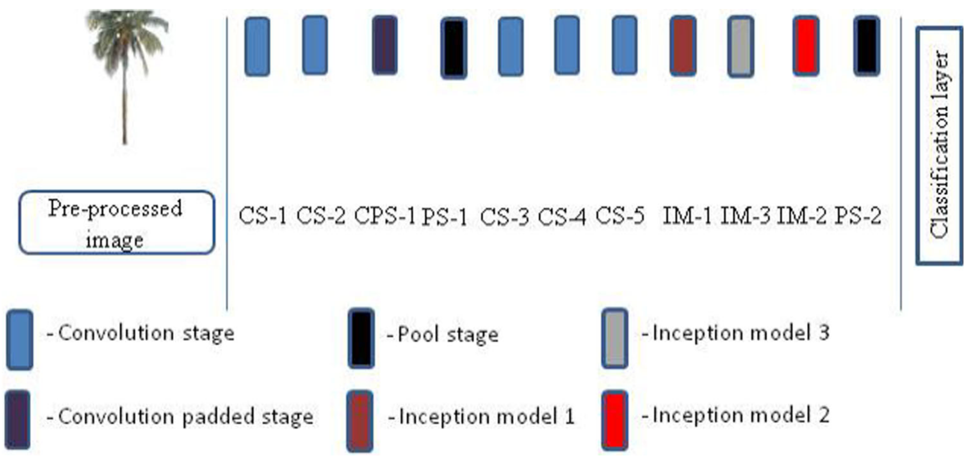

Fig. 4 Inception net framework

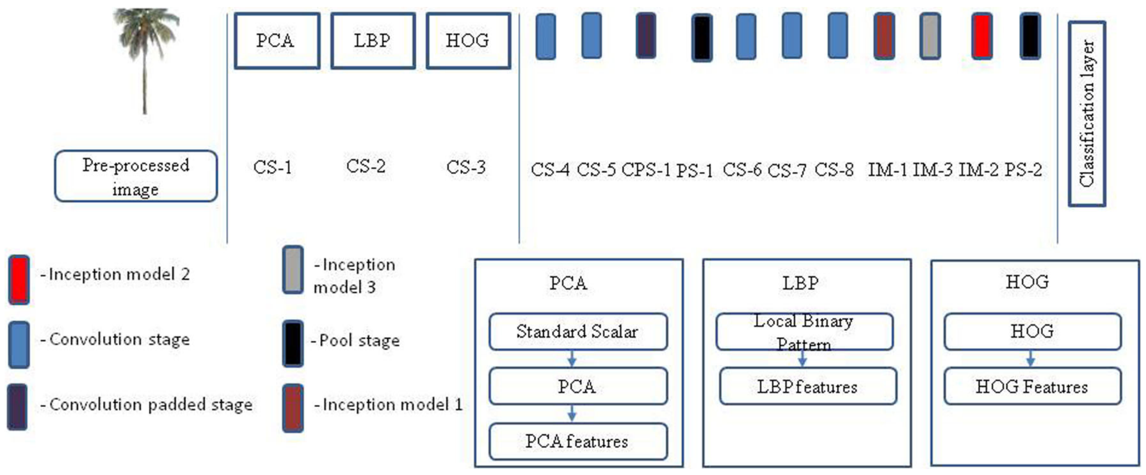

Fig. 5 Framework of modified inception net for feature extraction

extracted individually and multiplied by generalized scaling factor and added together to make it gray scale image.

$$S = \left[N(GL) - \sum_{i=1}^{N} GL_i\right] * \left[N * \sum_{i=1}^{N}\left(GL_i - \left[\sum_{i=1}^{N} GL_i\right]\right)^2\right]^{\frac{1}{2}} \quad (1)$$

Standardization of gray scale image data should be done to ensure that all data points come under same scale. Equation (1) is used to standardize in our case. Eigen values are sorted from ascending to descending order and J will store 'n' values from the beginning of the sorted data.

$$\forall k = 1 : mPCo_k = [J_k.T] * S \quad (2)$$

Equation (2) helps to find the principal component values by multiplying the eigen vector (J_k) which is related to that eigen value with the standardized data(S).

4.2 Mathematical modeling of hyper tuning support vector machine

In this subsection, we presented the equations for w_x, w_y, α and b. These equations are used to find the time difference between traditional SVM method and our proposed model. Features that are extracted using modified inception net are considered as feature array to find hyper plane equations.

$\forall i = 1 : mD = \{x_i, y_i\}$

$$H_i = \begin{cases} {w_1}^T * x + b_1 \geq 0; y_1 = i - 1 \\ {w_1}^T * x + b_1 < 0; y_1 = i + 1 \\ . \\ {w_2}^T * x + b_2 \geq 0; y_2 = i - 1 \\ {w_2}^T * x + b_2 < 0; y_2 = i + 2 \\ . \\ . \qquad \forall i = 0 : n \\ . \\ . \\ . \\ . \\ {w_n}^T * x + b_n \geq 0; y_2 = n + 1 \\ {w_n}^T * x + b_n < 0; y_2 = n \end{cases} \tag{3}$$

Equation (3) contains 'n' hyper plane equations. Number of hyper planes required is based on user input. Each set of hyper plane has 2 equations. One equation helps to plot positive values of the hyper plane data on one side of hyper plane and second equation helps to plot on other side of hyper plane. Based on number of classes H_i projects hyper plane which divides two classes. In our case, dataset contains three different classes. So, 'n' should be set to three and three sets of hyper plane equations are calculated. Each set of hyper plane equations gives one hyper plane. So, the whole dataset will be divided into three classes by using three different hyper planes

$$\Upsilon_i = \frac{y_i[{w_k}^T * x + b_k]}{||w_k||} \forall k = 1 : n, \forall i = 1 : m \tag{4}$$

Equation (4) is used to calculate the marginal distance. It is the distance between hyper plane to one of the nearest support vectors.

$$minmax(\alpha_k, w_k, b_k) = \frac{1}{2} * \left({w_k}^T * w_k\right) - \sum_{i=1}^{m} \alpha_i * y_i({w_k}^T * x_i + b_k) + \sum_{i=1}^{m} \alpha_i \forall k = 1 : n \tag{5}$$

$$\frac{\partial(minmax)}{\partial w_k} = \sum_{i=1}^{m} \alpha_i * x_i * y_i \tag{6}$$

$$\frac{\partial(minmax)}{\partial b_k} = \sum_{i=1}^{m} \alpha_i * y_i \tag{7}$$

Equation (5) is the minmax equation, here (w_k, b_k) are minimizing with maximizing α_k. From this equation we can get the equations for both weighted term (w_k) and bias term (b_k). To get the equation for weighted term differentiation of mixmax with respect to w_k should be done and that is shown in Eq. (6), similarly, b_k as shown in Eq. (7). To find the α, w (weight term), b (bias term) equations, exponential approximations is used. For this approach, we collected α, w, b values with Z as 142 and 100 samples separately. For 142 samples we got α = 121,699,445, $w_{(x, \quad y)}$ = (4.085,0.051), b = − 6.6380. For 100 samples α = 102,756,447, $w_{(x, \quad y)}$ = (− 0.0022, 0.00025), b = − 0.99 has been obtained.

$$\alpha(Z) = Ae^{-\gamma Z} \tag{8}$$

Equation (8) shows the exponential approximation used to find α, with A and γ are constants. To find A and γ, apply natural logarithm on both sides,

$$18.6 = ln(A) - 142\gamma \tag{9}$$

$$18.4 = ln(A) - 100\gamma \tag{10}$$

$$\alpha(Z) = 62409971.79e^{0.00476Z} \tag{11}$$

Solving Eqs. (9) and (10), we get exponential equation for α. To obtain α for any number of samples, Z should be substituted in Eq. (11).

$$w_x(Z) = Be^{-\beta Z} \tag{12}$$

Equation (12) helps to find w_x exponential approximation. Here B and β are the constants; from these constants w_x equation is calculated. By applying natural logarithm on both sides, we get,

$$1.40 = ln(B) - 142\beta \tag{13}$$

$$(-6.11 + 3.14i) = ln(B) - 100\beta \tag{14}$$

Solving Eqs. (13) and (14), we get exponential equation for w_x.

$$w_x(Z) = \left(3.3 * 10^{-7}\right) + (6.3 * 10^{-7}i)e^{(0.17+0.074i)Z} \tag{15}$$

$$w_y(Z) = Ce^{-\varepsilon Z} \tag{16}$$

$$w_y(Z) = (2.02 * 10^{-9})e^{0.12Z} \tag{17}$$

Following the same method to find w_x, constants C and ε are calculated for Eq. (16). Equation (17) is the exponential approximation of w_y.

$$b(Z) = D * (\frac{1}{2})^{Zx} \tag{18}$$

To find the equation for bias term, exponential approximation has been used which is mentioned in Eq. (18). Here D and x are the constants need to be found to frame equation for bias term.To find D and x from Eq. (18), apply natural logarithm on both sides,

$$(1.89 + 3.14i) = ln(c) - 98.4x \tag{19}$$

$$(-0.01 + 3.14i) = ln(c) - 69.3x \tag{20}$$

Solving Eqs. (19) and (20) we get exponential equation for b

$$b(Z) = -3981.25 * \left(\frac{1}{2}\right)^{0.065Z} \tag{21}$$

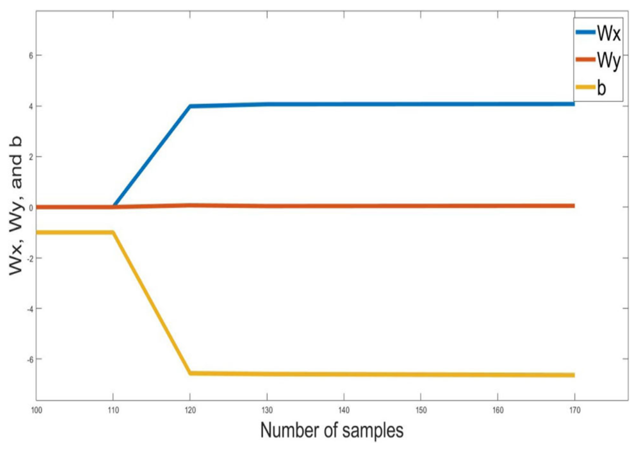

Fig. 6 Number of sample points Vs weighted and bias terms

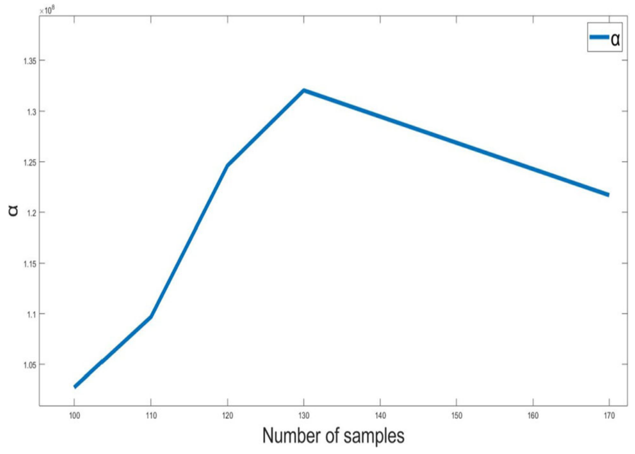

Fig.7 Number of sample points Vs α

After obtaining equations for α, *w* (weight term), and *b* (bias term), we can use these equations to find $minmax(\alpha_k, w_k, b_k)$. For these number of sample points, we get different values of α, *w*, and *b*.

Figure 6 shows the plot among number of sample points, weighted term, and bias term. The w_x value increases with the increase in number of samples given as input. As the w_x increases, graph shows decrement in the *b* values. W_y is very small value which is near to zero for all the samples taken. The number of sample points were mentioned on the X-axis. w_x, w_y, and *b* were calculated using Eqs. (15), (17), and (21), for different number of samples. On the Y-axis, w_x, w_y, and *b* values obtained for different sample points were plotted. Figure 7 shows the graph plotted between α and different sample points. For a high α the misclassification will be less. In Fig. 7, 130 samples have high alpha values and hence less misclassification. α is calculated using Eq. (11).

To obtain the results of time comparison, the dataset (without augmentation) is divided into five different smaller datasets as shown in Table 4. The original data contained measurements of height, inclination and orientation but the augmented data doesn't have any measurements. So, the dataset is created using original data and not augmented data. Number of samples for first dataset is 1/5th of original data. For, second dataset, the next 1/5th part of the orginal dataset is taken, and first dataset is also added to it. Similarly, 5 datasets are created with different number of samples as presented Table 4. Time taken for training by SVM model is very high as compared to MIN-SVM model. In all the five iterations MIN-SVM model gave output in less duration compared to SVM model. From the above results, we can conclude that our Hyper tuning SVM model is more time efficient than SVM model.

4.3 Mathematical modeling of Pareto probability distribution

We are representing *C* and gamma as probability distributions and accuracy, i.e., *f(x)* as a probability distribution function. Based on *C*, gamma and accuracy different probability distributions functions are calculatedand found Residual Sum of Squared (RSS), Loc (mean) and Scale (standard deviation) scores for all the distributions. Based on Least Value of RSS we selected Pareto distribution as best fit for our data than remaining probability distributions. *x* is an array which contains all the features that are extracted, and *y* is the target class. *C* and gamma parameters are influencing accuracy other than size of dataset.

Table 4 Comparison plot for time taken for training using SVM model and our model

Number of samples	Time taken by SVM model (secs)	Time taken by MIN-SVM model (secs)
157	0.1823	0.0146
313	0.2016	0.0154
469	0.217	0.0168
625	0.273	0.0234
787	0.639	0.051

Table 5 Comparison plot for time taken for training using gridsearch CV model and our model

Number of samples	Time taken for gridsearch CV(s)	Time taken using Pareto probability distribution model(s)
3000	832.648	10.329
6000	5323.139	58.6004
9000	11,683.19	109.01
12,000	21,992.783	180.173
17,000	40,731.797	322.41

$$F_1(x_1) = \Pr(C > x_1) = \begin{cases} x * y * \left(\left(\frac{x_{1m}}{x_1}\right) \wedge \alpha\right); & x_1 \geq x_{1m} \\ 1; & x_1 < x_{1m} \end{cases} \quad (22)$$

$$F_2(x_2) = \Pr(G > x_2) = \begin{cases} x * y * \left(\left(\frac{x_{2m}}{x_2}\right) \wedge \alpha\right); & x_2 \geq x_{2m} \\ 1; & x_2 < x_{2m} \end{cases} \quad (23)$$

Equation (22) and (23) represent Pareto Probability Distribution functions for both C and gamma. Where x_{1m} and x_{2m} stands for scale parameter that we obtain from standard deviation. α is the scale parameter, x_1 is the minimum value of C and x_2 is the minimum value of gamma. To find the best C and gamma values for our dataset, different combinations of C and gamma are tried, and accuracy is calculated. Accuracy keeps on changing for every combination of C and gamma, the accuracy for best C and gamma is the final accuracy of model.

Table 5 shows the time taken for training using GridSearchCV and our Pareto probability distribution model. In this case, we divided the whole dataset into 5 basedd on number of samples. The comparison shows Pareto probability distribution model can complete the test in lesser time as compared to GridSearchCV method. In each case MIN-SVM model gave fast results than GridSearchCV.

5 Results

The developed framework is trained and tested on a system configured as 11th Gen Intel® Core™ i7—11,700 @ 4 GHz clock frequency with 32 GB RAM and 8 GB GeForce RTX graphics card. To train and test the developed MIN-SVM model, "PyCharm" development environment is used. Accuracy, precision, recall, F1-score are the key metrics for every model to check its effectiveness. Accuracy can be calculated by comparing the original images (validation or testing images) with trained images. Receiver Operating Characteristics (ROC) curve is used to evaluate the developed Modified Inception Net Based

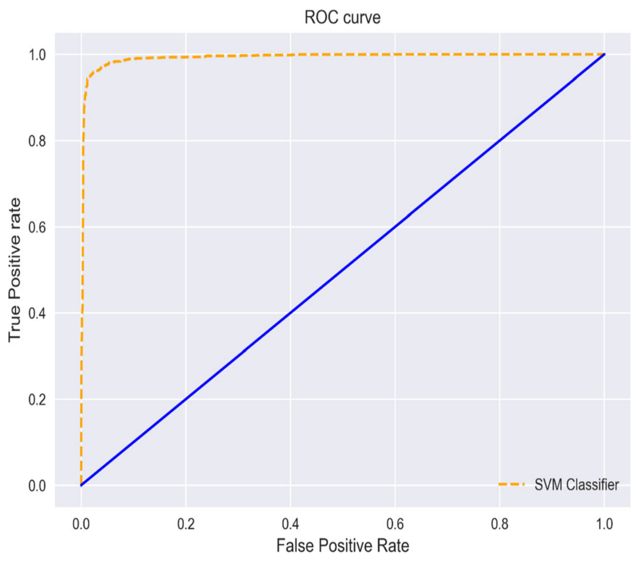

Fig. 8 Receiver Operating Characteristics curve of MIN-SVM model

Hyper Tuning Support Vector Machine (MIN-SVM). Grid Search Cross Validation is used to find the best estimator (C and gamma combination).

5.1 Receiver operating characteristics curve

ROC curves show the graphical representation of curves constructed between true positive rate and false positive rate of classification model. True positive rate is also termed as "sensitivity" and false positive rate is known as "specificity. After constructing ROC curve, Area Under Curve (AUC) is determined.

Figure 8 shows ROC of Hyper Tuning SVM classifier. The AUC of 0.991 has been achieved for the curve shown in Fig. 8. In general, AUC ranges from 0 to 1. AUC values closer to 1 indicates that prediction % is high. From Fig. 8, it can be concluded that the proposed MIN-SVM model can give high accuracy in classifying coconut trees based on height, inclination, and orientation.

The ROC plot of class a, class b, and class c is shown in Fig. 9. Plotting the ROC for all classes helps toknow which class is more efficient in classification. Figure 9 shows that class a, class b, and class c have almost same accuracy of 95%.

5.2 Hyper tuning support vector machine and hyper parameter plot

In the Hyper tuning SVM classifier, RBF is used as a kernel, C and gamma as hyperparameters. The gamma hyperparameter assists to decide boundary based on curvature weight. When the gamma value is high, the gamma considers the data points which are near to the decision

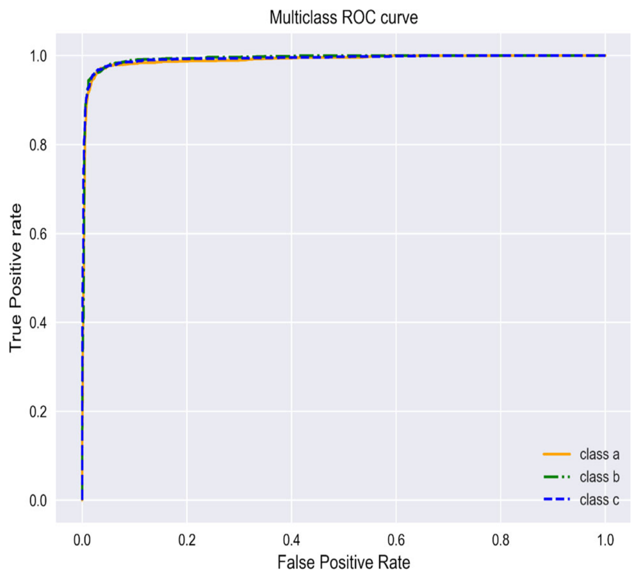

Fig. 9 Multi-class receiver operating characteristics curve of MIN-SVM model

boundary in the hyper plane. If the gamma value is low, the gamma considers the data points which are far from the decision boundary in the hyper plane. The second hyper parameter is the *C* parameter is used as a penalty parameter for the error data points. Regularization parameter controls the trade-off between the classification of training points and the decision boundary. SVM model chooses the data points as a support vector. When the *C* value is high, *C* choose more data points as a support vector which leads to overfitting. If *C* value is low, *C* considers very few data points as a support vector which leads to underfitting. So, optimal gamma and *C* parameter values must be chosen. We can find the optimal values of *C* and gamma parameters using GridSearchCV.

GridSearchCV works like an iteration process. The parameters used in this cross-validation are *C* and Gamma. Each iteration has different *C* and Gamma values, and accuracy is calculated by SVM for that particular *C* and gamma values. The combination of *C* and Gamma which yields high accuracy is considered as the best fit for the dataset. Best fit of *C* and gamma does not affect either by overfitting or underfitting our data. The *C* and Gamma values not only influence accuracy but also Variance and bias. When the gamma value and *C* values are high, the bias becomes high. On contrary, if the gamma value and the *C* value are low, the bias becomes low.

Figure 10 shows the accuracy of *C* and Gamma combinations. The plot is divided into grids to represent each *C* and Gamma values. Based on *C* and Gamma combinations the accuracies are predicted, and one color is assigned to each combination. The accuracies of each *C* and gamma are marked in distinct colors across the graph as shown in

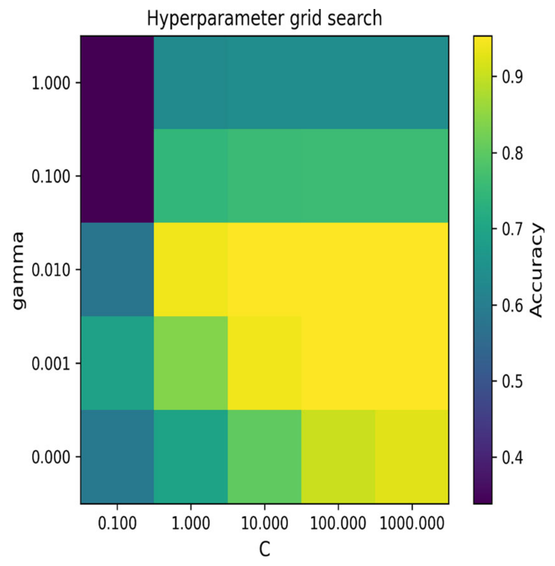

Fig. 10 Hyper parameter plot of MIN-SVM model

Fig. 10. Each color shows different accuracy levels that are mentioned on the right side of the figure. In Fig. 10, at *C* = 100 and gamma = 0.001, we can see the highest accuracy of 95.35%. GridSeachCV model helps to find best fitting (*C* and gamma combination) for MIN-SVM algorithm.

5.3 Change in C values

Table 6 shows the accuracy measured for *C* = [0.1, 1, 10, 100, 1000] and gamma = 0.001 combination. From GridSearchCV, the best parameters (fitting) obtained is (*C* = 100 and gamma = 0.001) with 95.35% accuracy. In Table 7, it is clearly shown that (*C* = 0.1 and gamma = 0.001) combination has minimum accuracy of 69.17%. The accuracy increased with increase in *C* values, but for (*C* = 1000 and gamma = 0.001) combination there is a slight decrease of 0.11% accuracy.

Figure 11 shows the plot between precision, recall, f1-score and *C* parameter. Since all the values obtained for precision, recall, and f1-score were same, three graphs were aligned. Figure 11 clearly shows that (*C* = 100 and gamma = 0.001) combination has highest precision, recall, and f1-score of 95.35%. CPU usage and RAM usage of MIN-SVM model are shown in Fig. 12. (*C* = 0.1 and

Table 6 Accuracy for different values of C with gamma = 0.001

C	Accuracy (%)
0.1	69.17
1	84.03
10	94.13
100	95.35
1000	95.23

Table 7 Variables and its abbreviations used in mathematical modeling of MIN-SVM

Variable	Abbreviation
x_i	Features of an image
y_i	Target class
w_k	Weighted term
b_k	Bias term
D	2-D Array contains x_i and y_i
α_k	Lagrange multiplier
H_i	Hyperplane parameter
Υ_i	Marginal distance
F(x)	Accuracy
Pr(x)	Pareto probability distribution
S	Standardization
I	Number of rows
N	Total number of columns present
J	Eigen values
G	Gray level image
C	RGB image
F	Feature array
Z	Number of samples
k	Number of principal component values required
n	Number of hyper planes required
M	Number of images in the dataset

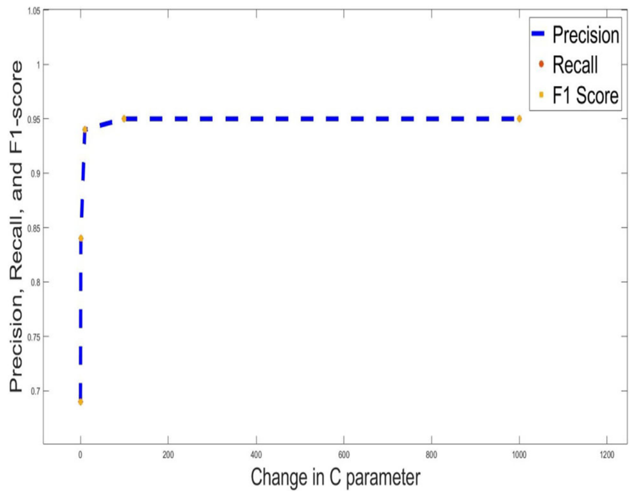

Fig. 11 Precision, recall, and F1-score plot for different C values with gamma = 0.001

gamma = 0.001) combination used 11.75% of CPU while training. Whereas (C = 100 and gamma = 0.001) combination used 12.2% of CPU for training. 6.24% of RAM is used by (C = 0.1 and gamma = 0.001) combination. Whereas (C = 1000 and gamma = 0.001) combination used 7.74% of RAM. The best combination (C = 100 and gamma = 0.001) has a RAM usage of 7.47%. In Fig. 12, we presented a plot between CPU usage, RAM usage, and C parameter.

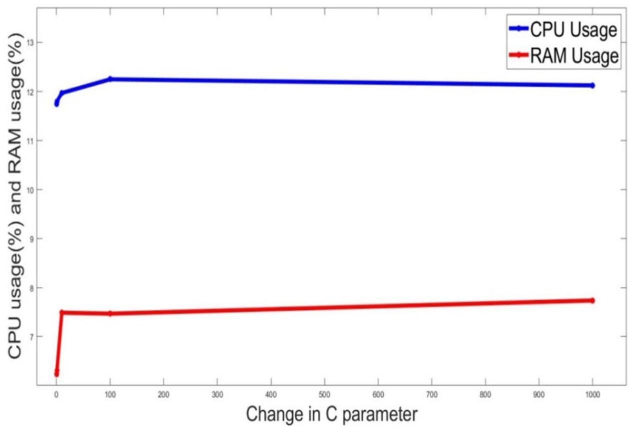

Fig. 12 CPU usage and RAM usage plot for different C values with gamma = 0.001

Computation time is an important factor to consider while testing or training a model. In Table 8, we presented the time required for each combination of C with gamma = 0.001. Lowest computation time is achievedby (C = [100, 1000] and gamma = 0.001) combination. Highest time for training is consumed by (C = 0.1 and gamma = 0.001) combination.

5.4 Change in gamma values

Table 9 shows the accuracy measured for gamma = [1, 0.1, 0.01, 0.001, 0.0001] and C = 100 combination. In Table 9, it is clearly shown that (gamma = 1 and C = 100) combination has minimum accuracy of 64.11%. The highest accuracy of 95.35%is achieved by (gamma = 0.001 and C = 100) combination. The accuracy increased with decrease in gamma values, but for (gamma = 0.0001 and C = 100) combination there is a 4.9% decrease in accuracy.

The results of precision, recall, and f1-score for different gamma values with C = 100 is presented in Fig. 13. Lowest precision value of 0.64 is obtained for (gamma = 1 and C = 100) combination. Whereas the highest accuracy of 0.95 is achieved by (gamma = [0.01, 0.001] and C = 100) combination.The lowest recall value of 0.82 is obtained for (gamma = 1 and C = 100) combination.

Table 8 Time taken for testing each fitting for different C values with gamma = 0.001

C	Computation time (sec)
0.1	296.4
1	228
10	184.8
100	178.8
1000	177.6

Table 9 Accuracy for different values of gamma with C = 100

Gamma	Accuracy (%)
1	64.11
0.1	75.91
0.01	95.21
0.001	95.35
0.0001	90.45

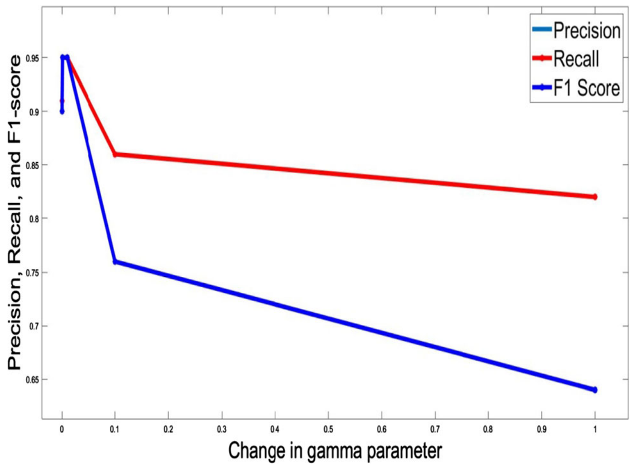

Fig. 13 Precision, recall, and f1-score plot for change in gamma values with C = 100

Whereas the highest accuracy of 0.95 is achieved by (gamma = [0.01, 0.001] and C = 100) combination.

Figure 13 shows the plot between precision, recall, f1-score and gamma parameter. Figure 13 clearly shows that (gamma = 0.001 and C = 100) combination has highest precision, recall, and f1-score of 95.35%.

The plot between CPU usage, RAM usage, for change in gamma values with C = 100 are presented in Fig. 14. CPU usage and RAM usage are calculated gamma = [1, 0.1, 0.01, 0.001, 0.0001] and C = 100 combinations are shown in Fig. 19. (gamma = 1 and C = 100) combination resulted in CPU usage of 12.47% for training. On contrary, (gamma = 0.01 and C = 100) combination ended up in CPU usage of 9.9% only. The (gamma = 1 and C = 100) combination used only 6.71% of RAM. On the other hand, 8.08% of RAM is used by (gamma = 0.0001 and C = 100) combination. Best fitting of MIN-SVM model (gamma-0.001 and C-100) combination used 10.11% CPU and 7.59% RAM.

In Table 10, computation time is measured. Computation times were measured for gamma = [1, 0.1, 0.01, 0.001, 0.0001] and C = 100 combinations, while testing MIN-SVM model. (gamma = 0.01 and C = 100) combination took more time for training. Whereas (gamma = 0.0001 and C = 100) combination took less time for training.

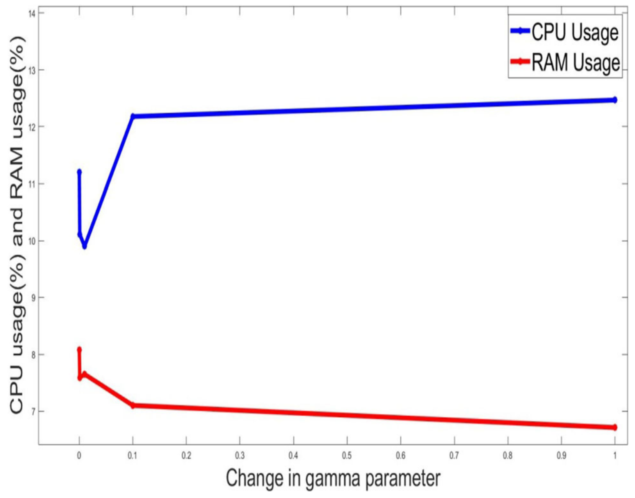

Fig. 14 CPU usage and RAM usage plots for different gamma values with C = 100

5.5 Change in kernels

Table 11 shows the accuracy comparison for different kernels used with MIN-SVM model. The highest accuracy of 95.35% is achieved by RBF kernel. Sigmoid kernel scored less accuracy of 84.45%. Table 11 justifies the selection of RBF kernel for training the proposed MIN-SVM mode. Figure 15 shows graphical representation of Precision, Recall, and F1-scores for different kernels. RBF kernel has achieved highest precision, recall, and f1-scores. Sigmoid kernels scored lowest Precision, Recall, and F1-scores.

Figure 16 shows the graphical representation of CPU usage, RAM usage for four different kernels. RBF kernel and linear kernel has less CPU usage compared to poly and sigmoid kernels. Sigmoid kernel and poly kernel has high RAM usage compared to RBF kernel and linear kernel.

Table 12 shows the computation time required for testing MIN-SVM with different kernels. The computation time of linear kernel is very low and RBF kernel took more time. Although time taken for testing is high for RBF kernel, it scored good accuracy of 95.35%. For each kernel we used 125 fittings to train SVM model.

Table 10 Computation time required for testing for different gamma values with C = 100

gamma	Computation time (sec)
1	387.6
0.1	382.8
0.01	463.2
0.001	205.2
0.0001	175.2

Table 11 Accuracy comparison for different Kernels

Kernel	Accuracy (%)
Radial Basis Function	95.35
Sigmoid	84.45
Linear	89.53
Poly	93.45

Table 12 Computation time used by different kernels while testing

Kernel	Computation time (sec)
Radial Basis Function	324.56
Sigmoid	318.33
Linear	136.72
Poly	320.54

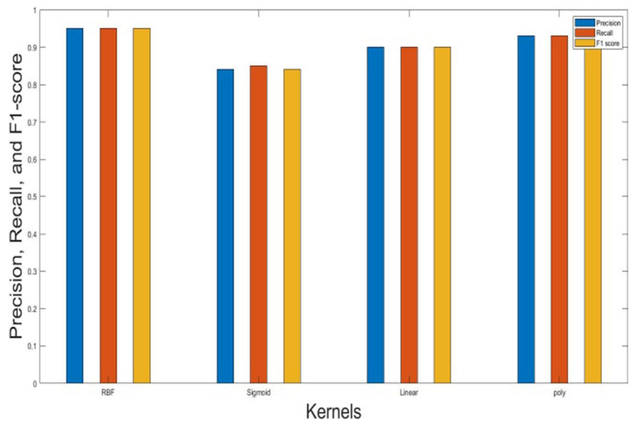

Fig. 15 Precision, recall, and F1-score plotfor different kernels

6 Bench marking

To check the effectiveness of our proposed algorithm, we trained and tested our dataset with three different existing algorithms. VGG Net [30], ResNet [31], and InceptionNet [32], algorithms are used for comparison. For this approach, we measured Accuracy, F1-score, Recall, Precision, CPU usage, RAM usage, and computational time for each algorithm in testing and training datasets. The details of the model are given in Table 13. All the methods (including the proposed method) are trained with the same dataset, training and testing are performed in same hardware. not doing any tuning (tuning of parameters) to the proposed method specifically to this problem.

In Table 13, parameters involved in algorithms are listed. For all algorithms same Activation function, Optimizer, Loss and metrics were used. But the number of layers and functions are different for every algorithm. VGG has minimum of 18 layers and 4 functions, whereas MIN-SVM has as maximum 51 layers and 7 functions.

7 Discussion

Table 14 shows the Accuracy, Precision, Recall, F1-score, CPU usage, RAM usage, and Computation time of different algorithms. Traditional CNN models named VGG Net, Inception Net, and ResNet are used for comparison with our MIN-SVM model. In Table 14, the computation time means the time taken for the testing of the 5100 images (30% of the total dataset). From Table 14, it is clear that MIN-SVM classification acquired better accuracy compared to other algorithms. For VGG Net, Inception Net, and ResNet, 20 epochs and batch size of 1458 is used for training and testing. For all the models, same dataset of

Fig. 16 CPU usage and RAM usage for different kernels

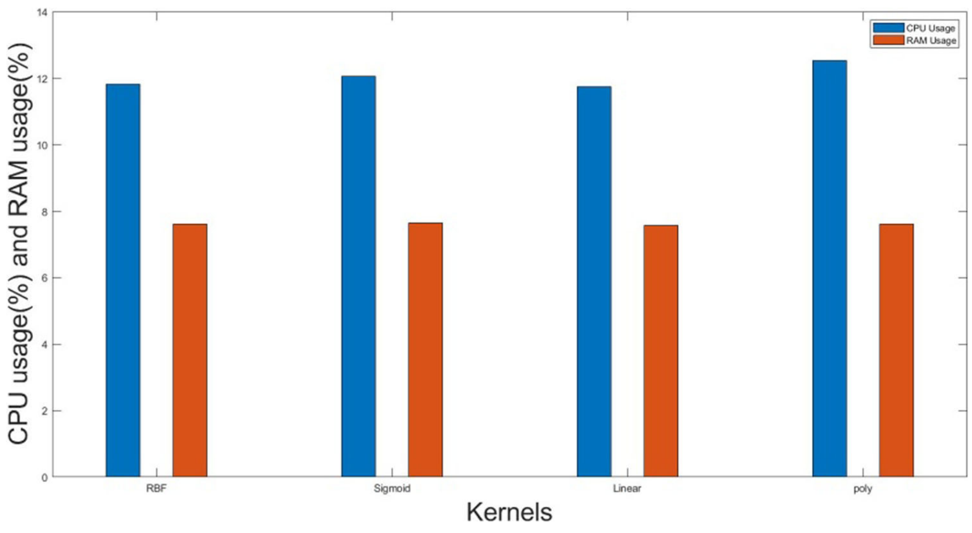

Table 13 Parameter used in comparison algorithms

Parameters	VGG Net	Resnet	Inception Net	MIN-SVM
Image size	224, 224, 3	224, 224, 3	224, 224, 3	224, 224, 3
No of Layers	18	50	48	51
No of Functions	4	6	7	7
Optimizer	Adam	Adam	Adam	Adam
Loss	CCE	CCE	CCE	CCE
Metrics	Accuracy	Accuracy	Accuracy	Accuracy

Table 14 Performance evaluation of different algorithms

Comparison algorithms	Accuracy (%)	Precision	Recall	F1-score	CPU usage (%)	RAM usage (%)	Computation time (sec)
MIN-SVM	95.35	0.95	0.95	0.95	17.995	6.28	174
VGG Net	91.90	0.91	0.91	0.90	92.9	5.35	968
ResNet	71.95	0.71	0.71	0.71	93.65	5.45	568
Inception Net	81.66	0.81	0.82	0.81	95.30	5.53	340

seventeen thousand images is used for training and testing. The CPU and RAM usage for traditional CNN models are very high compared to our proposed MIN-SVM model. For traditional CNN models the CPU usage stands around 92%, but for MIN-SVM model CPU usage is only 17.99%. The computation time of MIN-SVM algorithm is also less compared to other CNN models. From the results presented in Table 14, proposed MIN-SVM model performs better in all aspects that are considered for model evaluation.

8 Conclusion

A Modified Inception Net based Hyper Tuning Support Vector Machine (MIN-SVM) classification method is proposed in this research work for coconut tree classification based on three morphological parameters including height, inclination and orientation. The MIN-SVM model performs better than traditional CNN models for feature extraction of Coconut trees. When using traditional CNN models more pre-processing works need to be done as compared to MIN-SVM model. The SVM classifier together with the Inception Net feature extractor gave better result as compared to other CNN models. In fact, the best result in this research is, when using Modified Inception Net for feature extractor and the Hyper tuning SVM as classifier, achieving a testing accuracy of 95.35%.

Additionally, the performance of MIN-SVM is compared with different models. The validation result proved that the MIN-SVM model can be used for coconut trees classification based on height, inclination, and orientation. As a future work of this research, we would like to include more parameters like pest diseases, soil condition, wind speed, wind directions, etc. We would also train the extracted features with different machine learning algorithms like Random Forest, KNN, Decision tree, etc.

9 Future research

As part of future work, we are planning to use the current morphology-based classification which will be helpful for categorizing the coconut trees from which coconuts can be harvested by the robotic coconut tree climber [4]. The proposed research work is trained and tested with a particular variety of coconut trees called West Coast Tall (WCT) coconut trees. This can be extended to the other varieties of coconut trees like Chowghat Orange Dwarf (COD) and Chowghat Green Dwarf (CGD) as part of future research.

Acknowledgements We want to express our ardent appreciation to Almighty God for furnishing us with the chance to finish the study effectively. We want to offer our thanks to Amrita Vishwa Vidyapeetham and the Humanitarian Technology (HuT) Lab for helping us in this work.

Data availability Moreover, the datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors would like to declare that there is no conflict of interest with this article.

References

1. Singh UP, Chouhan SS, Jain S, Jain S (2019) Multilayer convolution neural network for the classification of mango leaves infected by anthracnose disease. IEEE Access 7:43721–43729. https://doi.org/10.1109/ACCESS.2019.2907383
2. Windrim L, Carnegie AJ, Webster M, Bryson M (2020) Tree detection and health monitoring in multispectral aerial imagery and photogrammetric pointclouds using machine learning. IEEE J Sel Top Appl Earth Obs Remote Sens 13:2554–2572. https://doi.org/10.1109/JSTARS.2020.2995391
3. Zhu S, Zhang J, Shuai G, Hongli L, Zhang F, Dong Z (2020) Autumn crop mapping based on deep learning method driven by historical labelled dataset. IEEE Int Geosci Remote Sens Symp. https://doi.org/10.1109/IGARSS39084.2020.9323897
4. Megalingam RK, Darla VP, Nimmala CSK, Sankardas KS (2022) Computer vision-based measuring method to estimate the diameter of the coconut tree trunk. Int Conf Adv Technol 2022:1–6. https://doi.org/10.1109/ICONAT53423.2022.9725999
5. Megalingam RK et al (2021) Amaran: an unmanned robotic coconut tree climber and harvester. IEEE/ASME Trans Mechatron 26(1):288–299. https://doi.org/10.1109/TMECH.2020.3014293
6. Fernández R, Montes H, Surdilovic J, Surdilovic D, Gonzalez-De-Santos P, Armada M (2018) Automatic detection of field-grown cucumbers for robotic harvesting. IEEE Access 6:35512–35527. https://doi.org/10.1109/ACCESS.2018.2851376
7. Tufail M, Iqbal J, Tiwana MI, Alam MS, Khan ZA, Khan MT (2021) Identification of tobacco crop based on machine learning for a precision agricultural sprayer. IEEE Access 9:23814–23825. https://doi.org/10.1109/ACCESS.2021.3056577
8. Korkut UB, Göktürk ÖB, Yildiz O (2018) Detection of plant diseases by machine learning. In: 2018 26th Signal processing and communications applications conference (SIU). pp 1–4. https://doi.org/10.1109/SIU.2018.8404692
9. Venkataraman D, Mangayarkarasi N (2017) Support vector machine-based classification of medicinal plants using leaf features. In: 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI). pp 793–798. https://doi.org/10.1109/ICACCI.2017.8125939.
10. Medar R, Rajpurohit VS, Shweta S (2019) Crop yield prediction using machine learning techniques. In: 2019 IEEE 5th International conference for convergence in technology (I2CT). pp 1–5. https://doi.org/10.1109/I2CT45611.2019.9033611
11. Shruthi U, Nagaveni V, Raghavendra BK (2019) A review on machine learning classification techniques for plant disease detection. In: 2019 5th International conference on advanced computing & communication systems (ICACCS). pp 281–284. https://doi.org/10.1109/ICACCS.2019.8728415
12. feng S, zhao J, liu T, zhang H, zhang Z, guo X (2016) Crop type identification and mapping using machine learning algorithms and sentinel-2 time series data. IEEE Access 12:3295–3306. https://doi.org/10.1109/JSTARS.2019.2922469
13. Alfred R, Obit JH, Chin CP-Y, Haviluddin H, Lim Y (2021) Towards paddy rice smart farming: a review on big data, machine learning, and rice production tasks. IEEE Access 9:50358–50380. https://doi.org/10.1109/ACCESS.2021.3069449
14. Chaitanya Reddy P, Chandra RMS, Vadiraj P, Ayyappa Reddy M, Mahesh TR, Sindhu Madhuri G (2021) Detection of plant leaf-based diseases using machine learning approach. In: 2021 IEEE International conference on computation system and information technology for sustainable solutions (CSITSS). pp 1–4. https://doi.org/10.1109/CSITSS54238.2021.9683020
15. Jaisakthi SM, Mirunalini P, Thenmozhi D, Vatsala (2019) Grape leaf disease identification using machine learning techniques. In: 2019 International conference on computational intelligence in data science (ICCIDS). pp 1–6. https://doi.org/10.1109/ICCIDS.2019.8862084
16. Treboux J, Genoud D (2018) Improved machine learning methodology for high precision agriculture. Glob Internet Things Summit (GIoTS) 2018:1–6. https://doi.org/10.1109/GIOTS.2018.8534558
17. Memon R, Memon M, Malioto N, Raza MO (2021) Identification of growth stages of crops using mobile phone images and machine learning. In: 2021 International conference on computing, Electronic and Electrical Engineering (ICE Cube). pp 1–6. https://doi.org/10.1109/ICECube53880.2021.9628197
18. Jena L, Behera SK, Sethy PK (2021) Identification of wheat grain using geometrical feature and machine learning. In: 2021 2nd International conference for emerging technology (INCET). pp 1–6. https://doi.org/10.1109/INCET51464.2021.9456281
19. Daneshwari AN, Basavaraju DR (2022) Corn leaf image classification based on machine learning techniques for accurate leaf disease detection. Int J Electric Comput Eng 12(3):2509
20. Hassan SM, Maji AK (2022) Plant disease identification using a novel convolutional neural network. IEEE Access 10:5390–5401. https://doi.org/10.1109/ACCESS.2022.3141371
21. SCK, Jcd, Patil N (2022) Cardamom plant disease detection approach using efficientnetV2. IEEE Access 10:789–804. https://doi.org/10.1109/ACCESS.2021.3138920
22. Sangeevan S (2021) Deep learning-based pesticides prescription system for leaf diseases of home garden crops in Sri Lanka. Int Res Conf Smart Comput Syst Eng 2021:94–98. https://doi.org/10.1109/SCSE53661.2021.9568308
23. Tunio MH, Jianping L, Butt MHF, Memon I (2021) Identification and classification of rice plant disease using hybrid transfer learning. In: 2021 18th International computer conference on wavelet active media technology and information processing (ICCWAMTIP). pp 525–529. https://doi.org/10.1109/ICCWAMTIP53232.2021.9674124
24. Ortega Cisneros S, Ruiz Varela JM, Rivera Acosta MA, Rivera Dominguez J, Moreno Villalobos P (2022) Pollen grains classification with a deep learning system GPU-trained. IEEE Latin America Trans 20(1):22–31. https://doi.org/10.1109/TLA.2022.9662170
25. Amin H, Darwish A, Hassanien AE, Soliman M (2022) End-to-end deep learning model for corn leaf disease classification. IEEE Access 10:31103–31115. https://doi.org/10.1109/ACCESS.2022.3159678
26. Weyler J, Milioto A, Falck T, Behley J, Stachniss C (2021) Joint plant instance detection and leaf count estimation for in-field plant phenotyping. IEEE Robot Autom Lett 6(2):3599–3606. https://doi.org/10.1109/LRA.2021.3060712
27. Lakshmanarao A, Babu MR, Kiran TSR (2021) Plant disease prediction and classification using deep learning convnets. Int Conf Artif Intell Mach Vis (AIMV) 2021:1–6. https://doi.org/10.1109/AIMV53313.2021.9670918
28. Trivedi NK, Maheshwari S, Anand A, Kumar A, Rathor VS (2023) Identify and classify corn leaf diseases using a deep neural network architecture. In: Proceedings of seventh international congress on information and communication technology (pp 873–880). Springer, Singapore
29. Ganeshkumar M, Sowmya V, Gopalakrishnan EA, Soman, KP (2022) Unsupervised deep learning approach for the identification of intracranial haemorrhage in CT images using PCA-Net and K-Means algorithm. In: Intelligent vision in healthcare (pp. 23–31). Springer, Singapore
30. Campos-Leal JA, Yee-Rendón A, Vega-López IF (2022) Simplifying VGG-16 for plant species identification. IEEE Lat Am Trans 20(11):2330–2338. https://doi.org/10.1109/TLA.2022.9904757

31. Hu W-J, Fan J, Du Y-X, Li B-S, Xiong N, Bekkering E (2020) MDFC–ResNet: an agricultural IoT system to accurately recognize crop diseases. IEEE Access 8:115287–115298. https://doi.org/10.1109/ACCESS.2020.3001237
32. Chen J, Chen W, Zeb A, Yang S, Zhang D (2022) Lightweight inception networks for the recognition and detection of rice plant diseases. IEEE Sens J 22(14):14628–14638. https://doi.org/10.1109/JSEN.2022.3182304