SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V11I1P110

Volume 11 Issue 1, 131-142, January 2024

Original Article

Texture Features-Based Detection of Plant Leaf Diseases Using RM-SVM

G. Nandha Kumar[1], V. Vijayakumar[2]

[1]*Department of Computer Science, Sri Ramakrishna College of Arts and Science, Tamilnadu, India.*
[2]*Department of Computer Science and Controller of Examinations, Sri Ramakrishna College of Arts and Science, Tamilnadu, India.*

[1]*Correspoding Author : mlgnandha@gmail.com*

Received: 18 October 2023 Revised: 28 November 2023 Accepted: 19 December 2023 Published: 13 January 2024

Abstract *- Texture features describe the spatial arrangement of pixels in an image. They can be used to categorize different forms of objects in an image, such as leaves, flowers, and fruits. In the context of plant leaf disease detection, texture features can be used to find the presence of diseases in leaves. In this paper, the proposed Machine Learning (ML) model can be used for texture feature-based detection of plant leaf diseases in a Redundant Multiclass Support Vector Machine (RM-SVM). RM-SVM is a Support Vector Machine (SVM) specifically designed for dealing with redundant features. RM-SVM is effective for texture feature-based detection of plant leaf diseases. However, RM-SVM can be computationally expensive to train and can be sensitive to the choice of hyperparameters. To reduce computational costs, K-Means clustering for redundancy removal is used. Finally, the Max-Rule is applied to the fuse score of predicted results to provide better accuracy at 95.6%.*

Keywords *- RM-SVM, K-Means clustering, Redundancy removal, Max-rule, Fusing score, Texture feature, Plant leaf disease detection.*

1. Introduction

Plant leaf diseases can be detected and categorized using ML. ML is a sort of Artificial Intelligence (AI) that enables systems to learn despite explicitly programming them. ML algorithms may be employed to analyze data and uncover trends from which forecasts can be made.

ML algorithms may be applied to analyze images of leaves to detect the presence of illnesses in the framework of plant leaf disease detection [1]. The algorithms can be trained on a data set of images of strong and contaminated leaves. Many ML techniques may be used to identify plant leaf disease. ML is a robust method to identify and categorize plant leaf diseases [2].

Once trained, the algorithms can organize original images of leaves and diseased ones. Machine learning algorithms are effective for detecting plant leaf diseases. SVMs are machine learning algorithms that may be utilized in regression and classification [3].

SVMs locate a hyperplane, which separates details into two distinct groups and categories. For binary classification, SVMs work by finding a hyperplane that separates the two classes of data points [4].

One-vs-one: This approach creates a separate SVM for each pair of classes. The SVM for a particular pair of classes tries to find a hyperplane that separates the two classes. The overall decision of the forecasts from the various SVMs is used to conclude [5].

It is used to classify data with a large number of classes. However, they can be computationally expensive to train, especially when the number of classes is significant [6]. Many different plant leaf diseases can be affected by various factors, including bacteria, fungi, viruses, and insects. Some of the most common plant leaf diseases [7, 8] include:

Bacterial leaf spot: This disease is caused by bacteria and is characterized by small, brown spots on the leaves. The adverts may eventually join to form larger, dead areas on the leaves.

Fusarium wilt: This disease is caused by a fungus and is characterized by wilting and yellowing of the leaves. The leaves may eventually fall off the plant.

Powdery mildew: This disease is affected by a fungus and is categorized by a white, chalky growth on the leaves. The growth may ultimately protect the entire leaf, preventing it from photosynthesizing.

Viruses: Viruses can cause various symptoms in plants, including yellowing, mottling, and curling of the leaves. Viruses can also cause stunting and death of plants.

Insect pests: Insects can also cause leaf diseases by feeding on the leaves and transmitting diseases. Some common insect pests that cause leaf diseases include aphids, scale insects, and spider mites.

The severity of a plant leaf disease can vary depending on the type of disease, the plant species, and the eco-friendly conditions [9]. Some diseases, such as bacterial leaf spot, can be relatively minor and do not cause significant damage to the plant. Other diseases, such as Fusarium wilt, can be severe and kill the plant [10].

However, despite the advancements in disease detection methodologies, there still exists a notable research gap in effectively leveraging the discriminative power of texture features, particularly in the context of plant leaf disease detection.

This research aims to enhance plant disease detection's accuracy, reliability, and efficiency by integrating robust texture feature extraction methods with an optimized SVM framework. The primary focus will be utilizing sophisticated texture descriptors and a well-tuned RM-SVM classifier to discern subtle variations in leaf textures caused by different diseases. This will facilitate early and precise disease identification.

This research endeavors to contribute significantly to the field by improving the accuracy of disease detection and offering a scalable and adaptable framework that could be implemented in automated agricultural systems. The ultimate goal is to empower farmers and agricultural practitioners with an effective tool for early disease diagnosis, enabling timely intervention and optimized disease management strategies. Through this innovative approach, the study aspires to fill the existing research gap and pave the way for more efficient and reliable plant disease detection systems.

2. Related Works

Most of our nation's inhabitants live in the agricultural sector. If any illness has impacted a plant for a prolonged period, then there is a shortfall in productivity in agricultural production. Because of this, it is essential to identify and investigate the illness. The approach described here is intended to organize the diseases in the tea leaf plantation [11]. Even if there is a significant rise in the population, agriculture ensures that all people will have access to food. In agriculture, detecting diseases that affect plants early is essential to ensure that there will always be enough food for everyone. It has been hypothesized that plant diseases may be predicted in their early stages [12]. However, it is unlucky to be able to forecast illnesses at such an early stage in the growth of the crops.

A unique content-based image retrieval system has been built in this study to categorize the illnesses that may affect tomato plant leaves [13]. The technique that has been presented separates similar images based on the color, shape, and texture of the tomato leaf. Steps need to be taken to ensure the highest possible yield from a crop to ensure the highest possible yield from a crop and eliminate diseases that affect rice plants. In the approach [14] that has been suggested, the updated K-means segmentation methodology is used to differentiate the targeted area from the background image of the rice plant.

Using the ELM deep learning algorithm, the author of this work offers a new method for the automated identification and categorization of plant leaf diseases [15]. Recent image and data analysis advances have made it easier to resolve various agricultural issues. Crop disease categorization and detection are critical for the technological and financial prosperity of the agricultural business [16]. Because the indications of infections are most readily apparent on the leaves of plants, leaves are the primary part of the plant that is examined to diagnose and identify illnesses. Detecting the disease by visual inspection is a complex undertaking in and of itself, and it calls for a significant amount of human knowledge [17].

A grayscale version of the image was first created, and then an overlapping sliding window was used to segment it into individual blocks. After obtaining the grey-level co-occurrence matrices from these blocks, six Haralick texture features were calculated on the image [18].

A technique that identifies illnesses from diseased apple leaves using a combination of machine learning and image processing ideas has been developed and presented in this research [19].

This study aimed to develop, via the utilization of segmenting images and a multiclass support vector machine. This system would allow for the early identification of illnesses that affect potato leaf tissue. This research endeavor starts with extracting leaf spots using the Thresholding approach [20]. The results of segmenting an image are retrieved depending on the colors' properties. Suggest a method for identifying illnesses by analyzing images of diseased apple leaves, using a combination of ideas from machine learning and image processing. This method [21] effectively differentiates apple leaves that have been infected and those that have not been affected.

A significant decline in terms of both the level of quality and the number of items produced by agriculture might be brought about by the presence of plant diseases. To keep agricultural yields high, automatic disease detection in plants is essential once the signs of the illness appear on the plant's leaves and fruit [22]. Plants are very susceptible to diseases,

which in turn indicate the development of the plant, which means the normal equilibrium of the agriculturalist. The incidence of contagions on various components of the houseplant, which are caused by multiple diseases, decreases crop production [23].

The conventional CNN models need a significant number of parameters, which results in an increased calculation cost-in this article [24] used depth separable convolution rather than regular convolution, which enabled a decrease in both the number of constraints and the amount of processing required.

The purpose of this study [25] is to give answers to farmers by focusing on the development of a method like this for use in agriculture, specifically for the detection of diseases in plant leaves at their earliest stages. The most important addition is a unique pixel replacement-based segmentation strategy and a double feature extraction method for improving the classification process.

Traditional machine learning models often struggle to effectively extract and utilize intricate texture information in plant leaves, limiting their diagnostic accuracy. Additionally, existing techniques' lack of robustness and scalability impedes their practical application in real-world agricultural settings. This study proposes a novel approach that harnesses Texture Features Detection of Plant Leaf Diseases using RM-SVM to address these limitations and bridge the existing research gap. RM-SVM addresses the limitations of traditional SVMs by enhancing robustness against outliers and improving classification performance in multiclass scenarios.

3. Proposed Model

Texture features based on detecting plant leaf diseases using redundant multiclass SVM (Support Vector Machine) is an innovative approach in plant pathology and agricultural research. This method leverages texture analysis techniques and the power of SVMs to accurately detect and classify different types of plant leaf diseases. An overall system model is shown in Figure 1.

The detection process begins with acquiring high-resolution images of plant leaves affected by diseases. These images are preprocessed to enhance their quality and remove any unwanted noise or artifacts that could interfere with the analysis. Next, texture features are extracted from the preprocessed leaf images. These features capture the unique patterns, structures, and pixel intensity variations of specific leaf diseases.

Once the texture features are extracted, a redundant multiclass SVM model is constructed. Unlike traditional SVMs designed for binary classification, redundant multiclass SVM extends SVMs to handle multiple classes. It achieves this by training several SVM classifiers, each prepared to distinguish between one class and the rest. The redundant multiclass SVM combines the results from these multiple classifiers to make the final prediction.

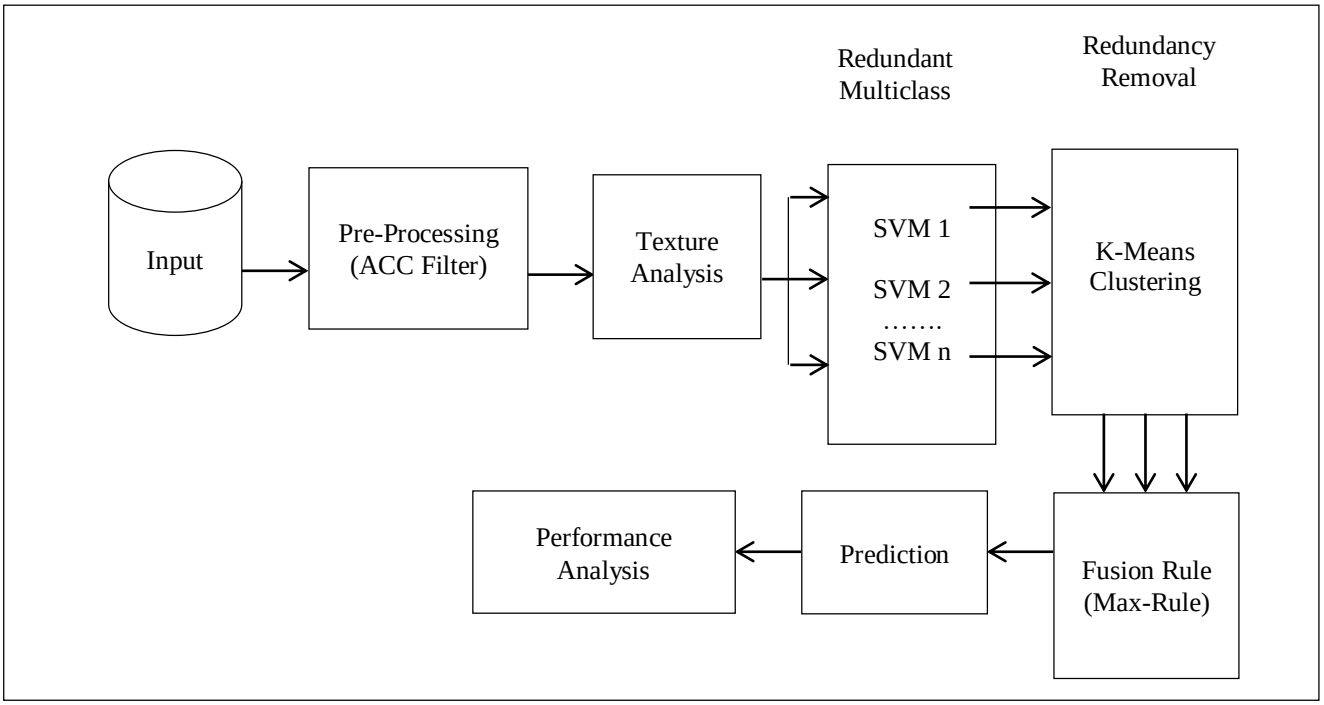

Fig. 1 Overall process of proposed model

3.1. Auto-Color Correlogram (ACC) Filter

The Auto-Color Correlogram (ACC) filter is used in processing images and machine learning techniques for analyzing the color distribution and spatial correlation of colors in an image. It was introduced by Huang et al. in 1997 as a color-based image retrieval method.

The ACC filter computes the color correlogram, representing the joint probability distribution of color pairs at different distances and color differences within an image. It captures both color content and spatial information, allowing for robust color-based image retrieval and recognition. Here's an overview of the steps involved in computing the Auto-Color Correlogram (ACC) filter.

3.1.1. Image Quantization

The first step is to quantize the image's color space into a finite number of color bins. This reduces the complexity of color analysis while preserving the general color distribution.

3.1.2. Color Correlogram Computation

The ACC filter calculates the probability of finding a neighboring pixel with a similar color within a specified distance for each color bin. The distance is defined in terms of pixels or a predefined spatial scale. The color correlogram is computed by counting the occurrences of color pairs with specific distances and color differences.

3.1.3. Normalization

After computing the color correlogram, it is normalized to ensure that the values lie within a specific range, typically between 0 and 1. This normalization helps compare and match images based on color correlogram representations.

The resulting ACC filter output is a compact illustration of an image's color distribution and spatial relationships. It can be used for various applications, such as content-based image retrieval, image indexing, and image recognition. By comparing the color correlograms of different images, one can measure the similarity or dissimilarity between images based on their color content and spatial distribution.

The Auto-Color Correlogram (ACC) filter has performed well in various image retrieval tasks, particularly in scenarios where color information plays a crucial role. It provides a robust and efficient way to represent and analyze the color characteristics of images, enabling effective image search and retrieval based on color similarity.

The mathematical model for the Auto-Color Correlogram (ACC) filter involves defining the color correlogram and its computation based on the joint probability distribution of color pairs at different distances and color differences within an image. Let's define the mathematical model for the ACC filter.

Image Quantization

Let N represent the number of color bins or quantization levels. Let C(x, y) denote the image's color value at pixel (x, y).

Color Correlogram

Define the ACC matrix as ACC (i, d, cd), where, i represents the color bin index, d represents the distance index, and cd represents the color difference index. ACC (i, d, cd) stores the joint probability of color bin i at distance d with color difference cd.

ACC Computation

Iterate over each pixel (x, y) in the image:

Obtain the current color value:

$$currentColor = C(x, y) \quad (1)$$

For each distance d and color difference cd.

Calculate the number of neighboring pixels at distance d from (x, y) with color difference within the threshold cd.

$$count = 0 \quad (2)$$

Iterate over each neighboring pixel (nx, ny) at distance d from (x, y):

If

$$abs(C(nx, ny) - currentColor) <= cd \quad (3)$$

Then increment the count by 1.

Assign the count to the corresponding ACC bin:

$$binIndex = findBinIndex(currentColor, colorBins) \quad (4)$$

$$ACC(binIndex, d, cd) += count \quad (5)$$

Normalization

Normalize the ACC matrix to ensure that the values represent probabilities or frequencies. Divide each element in the ACC matrix by the total number of considered pixel pairs.

Find Bin Index

The findBinIndex function maps a color value to its corresponding color bin index.

Iterate over the Color Bins

If the color value is lower than or greater than the contents of the current color bin, return the current index. If the color value exceeds the maximum bin color, assign it to the last bin index. This mathematical model outlines the computation of the

ACC filter, including the color correlogram representation and the steps for ACC computation and normalization.

Pseudocode for ACC Filter

```
function computeACC(image, colorBins, distances, color difference threshold):
  Initialize an empty ACC matrix of size [numBins, numDistances, numColorDifferences] with all elements set to zero
    for each pixel (x, y) in the image:
    currentColor = image(x, y)
        for each distance in distances:
      for each colorDifference in color Differences:
        count = 0
                  for each neighboringPixel (nx, ny) at distance d from (x, y):
            if abs(image(nx, ny) - currentColor) <= colorDifferenceThreshold:
                count += 1
                    binIndex = findBinIndex(currentColor, colorBins)
          ACC(binIndex, distance, colorDifference) += count
     Normalize ACC matrix by dividing each element by the total number of considered pixel pairs
    return ACC
function findBinIndex(color, colorBins):
  for index, binColor in enumerate(colorBins):
    if color <= binColor:
      return index
  return len(colorBins) - 1  // If color is greater than the maximum bin color, assign it to the last bin
```

In the pseudocode, image represents the input image, colorBins is an array defining the color quantization levels, distances is an array specifying the distances to consider, and colorDifferenceThreshold is the threshold range for color differences. The ACC matrix stores the computed ACC values.

The computeACC function iterates over each pixel in the image and calculates the ACC values for different distances and color differences. The findBinIndex function determines the index of the corresponding color bin for a given color value.

After computing the ACC values, the matrix is normalized by dividing each element by the number of considered pixel pairs, ensuring the values represent probabilities or frequencies.

3.2. Texture Analysis Technique

GLCM is a texture analysis technique in digital image processing and computer vision. GLCM extracts statistical information from an image to characterize its texture. Assuming we have an image represented as a grayscale matrix of size M x N, the steps for texture feature extraction using GLCM can be summarized as follows:

Step 1: Select the pixel distance (d) indicating the distance between the pixel pairs considered for co-occurrence calculation.
Step 2: Choose the angle or direction (θ) at which the co-occurrence pairs will be calculated. Common angles are 0°, 45°, 90°, and 135°.
Step 3: Create an empty co-occurrence matrix (GLCM) of size G x G, where G denotes the number of Grayscale stages within the image. (usually 256 for 8-bit grayscale images).
Step 4: Iterate over each pixel (i, j) in the image, excluding the boundary pixels.
Step 5: For each pixel, calculate the coordinates (i', j') of the neighboring pixel at distance d and angle θ.
Step 6: Increment the corresponding element (GLCM[i, j, p, q]) in the GLCM matrix, where p and q signify the gray levels of the pixels (i, j) and (i', j'), respectively.
Step 7: Divide each element of the GLCM matrix by the sum of all elements to obtain a probability matrix, ensuring that the values represent the probability of occurrence of a pixel pair at the specified distance and direction.
Step 8: Extract texture features: Compute various statistical measures from the GLCM, such as contrast, homogeneity, entropy, energy, and correlation, to describe the texture properties of the image.

3.2.1. Contrast (CON)

Calculate the contrast as the sum of their squared disparities in each element and the mean of the GLCM:

$$CON = \Sigma(i,j)[GLCM(i,j) * (i - j)^2] \quad (6)$$

3.2.2. Energy (or Angular Second Moment) (ASM)

Calculate the energy as the sum of the squared GLCM elements:

$$ASM = \Sigma(i,j)[GLCM(i,j)^2] \quad (7)$$

3.2.3. Homogeneity (or Inverse Difference Moment) (IDM)

Calculate the homogeneity as the sum of the GLCM elements divided by the absolute difference between their row and column indices:

$$IDM = \Sigma(i,j)\ [GLCM(i,j)\ /\ (1 + |i - j|)] \quad (8)$$

3.2.4. Entropy (ENT)

Calculate the entropy as the negative sum of the GLCM elements multiplied by their logarithms:

$$ENT = -\Sigma(i,j)\ [GLCM(i,j) * log(GLCM(i,j))] \quad (9)$$

3.2.5. Correlation (COR)

Calculate the correlation as the sum of the GLCM elements weighted by their row and column indices:

$$COR = \Sigma(i,j)\,[(i - \mu\_i) * (j - \mu\_j) * GLCM(i,j)] / (\sigma\_i * \sigma\_j) \quad (10)$$

Here, μ_i and μ_j are the means of the row and column indices, respectively, and σ_i and σ_j are the deviations of the row and column indices, respectively.

3.3. Redundant Multiclass Support Vector Machine (RM-SVM)

A Redundant MSVM is a type of SVM that can categorize data into multiple programs. MSVMs are typically used for difficulties where there are many classes or the classes are not well-separated. In a redundant MSVM, each data point is assigned to a single class. This means that no "borderline" data points could be classified as either class.

This can make MSVMs more accurate than other types of SVMs but also make them more computationally expensive. To reduce the computational complexity of redundant MSVMs, a technique called "redundancy removal" can be used. Redundancy removal involves removing data points that are not important for classification. This can be done by using various methods, such as k-means clustering.

3.4. K-Means Clustering for Redundancy Removal

The k-means clustering algorithm does not provide a mathematical model for redundancy removal in MSVM. However, we can describe the steps in applying k-means clustering for redundancy removal in MSVM. Here's an outline of the process:

Step 1 : Given a set of support vectors in the MSVM training set, denoted as X = {x1, x2, ..., xn}, where each support vector xi belongs to the feature space R^d.

Step 2 : Normalize or standardize the feature vectors, if necessary, to ensure they have a similar scale.

Step 3 : Select the number of clusters, K, for the k-means algorithm.

Step 4 : Initialize K cluster centroids (mean vectors) in the feature space, denoted as C = {c1, c2, ..., cK}.

Step 5 : Using a distance metric, most often the Euclidean distance, assign each support vector to the centroid that is located closest to it. This stage results in the formation of K clusters.

Step 6 : Update the centroids by computing each cluster's mean (centroid).

Step 7 : Repeat steps 5 and 6 iteratively until convergence, where convergence occurs when the assignments of support vectors to clusters either do not change appreciably anymore or have surpassed the maximum number of iterations established.

Step 8: After achieving convergence, analyze the resulting clusters to understand their characteristics.

Step 9: Select representative vectors from each cluster based on defined criteria. For example, you may choose the support vector closest to the centroid or the one with the highest classification confidence within each cluster.

Step 10: Update the support vector set by replacing the original support vectors with the selected representative vectors.

The mathematical models involved in the k-means clustering algorithm include computing the distance between support vectors and centroids (e.g., Euclidean distance) and updating the centroids by calculating the mean of the support vectors assigned to each cluster. Redundancy removal can improve the accuracy and speed of redundant MSVMs. Here are some of the advantages of using a redundant MSVM:

High accuracy: MSVMs are typically more accurate than other types of SVMs, such as binary SVMs.

Robust to noise: MSVMs are robust to noise, meaning they can still perform well even if the data contains some errors.

Flexible: MSVMs can classify data into any number of classes.

Overall, redundant MSVMs are a powerful tool for classification problems. However, they can be computationally expensive and sensitive to hyperparameters.

3.4.1. The Mathematical Model - RMSVM

Training Phase

Given a training dataset with C classes, let X = {x_1, x_2, ..., x_n} represent the feature vectors and Y = {y_1, y_2, ..., y_n} represent the corresponding class labels, where y_i belongs to the set {1, 2, ..., C}.

For each class i ($1 \le i \le C$):

Create a binary training set:

Define the binary labels

$$Z = \{z\_1, z\_2, \ldots, z\_n\} \quad (11)$$

Such that z_j = +1 if y_j = i and z_j = -1 otherwise.

The binary training set is denoted by X_i = {x_j | z_j = +1} and Z_i = {z_j | z_j = +1}.

Train a Binary SVM Classifier

Solve the binary classification problem using the binary training set X_i, Z_i to obtain the SVM hyperplane parameters w_i and b_i.

The decision function for class i is given by f_i(x) = sign(w_i · x + b_i), where · represents the dot product.

Store the trained classifier with the associated hyperplane parameters (w_i, b_i) for each class i.

Testing Phase
Given a test sample x_test, apply each binary classifier to obtain the distance or score:

Compute the distance or score for class i using f_i(x_test) = w_i · x_test + b_i.
Combine the distances or scores using a predefined fusion rule to determine the final predicted class:
The fusion rule can be based on max-rule, min-rule, or voting scheme, depending on the specific requirements.
For example, the predicted class can be determined as argmax_i(f_i(x_test)).

The RMSVM algorithm trains multiple binary SVM classifiers, one for each class, to create a redundant set of decision boundaries. During testing, the fusion rule combines the outputs of the binary classifiers to make the final prediction.

It's important to note that the specific implementation details of the SVM training, such as the choice of kernel function, regularization parameters, and optimization algorithms, can be adapted based on the requirements and characteristics of the dataset.

The mathematical model outlined above provides a general framework for understanding the RMSVM algorithm and its multiclass extension based on binary SVM classifiers.

min_{wt, bt} \frac{1}{2} ||wt||^2 + rb \sum_{i=1}^n \max_{j \neq y_i} \{1 - y_i (wt^Ty_i + bt) \} (12)

Where:
wt indicates the weight vector
bt signifies the bias term
rb is a regularization parameter
Ty is the ith training example
yi is the label of the ith training example

The objective function of the SVM minimizes the squared error among the predicted and true labels, while penalizing the model's complexity by adding a regularization term. The regularization term is weighted by the hyperparameter C, which controls the tradeoff between model complexity and accuracy. The SVM algorithm solves the optimization problem using a quadratic programming solver. The solution to the optimization problem is a set of support vectors, which are the training examples closest to the decision boundary. The decision boundaries are a hyperspace that divides data points into classifications.

The redundant multiclass SVM is a variant of the standard SVM that allows for multiple support vectors to be associated with each class. This can be useful for problems with many classes, as it can help reduce the complexity of the model.

3.4.2. Psuedocode for RM-SVM

```
function trainRMSVM(X, Y):
   numClasses = number of unique elements in Y
      classifiers = empty array of size numClasses
      for i = 1 to numClasses:
      binary labels = createBinaryLabels(Y, i)
      classifiers[i] = trainBinarySVM(X, binary labels)
return classifiers
function createBinaryLabels(Y, targetClass):
   binarylabels = empty array of size Y.length
 for i = 1 to Y.length:
      if Y[i] == targetClass:
         binaryLabels[i] = +1
      else:
         binaryLabels[i] = -1
    return binaryLabels
function trainBinarySVM(X, binaryLabels):
   // Train a binary SVM classifier using X as the feature matrix
and binaryLabels as the target labels
function predictRMSVM(X_test, classifiers):
   numClasses = classifiers.length
   scores = empty array of size numClasses
  for i = 1 to numClasses:
      scores[i] = classifiers[i].computeScore(X_test)
   // Combine scores using a fusion rule
   predictedClass = argmax(scores)
      return predictedClass
```

In the pseudocode, X represents the training feature matrix, Y represents the corresponding class labels, and X_test represents the feature matrix of the test sample. The train RMSVM function trains the RMSVM by creating binary labels for each class and training binary SVM classifiers. The predict RMSVM function uses the trained classifiers to compute scores for the test sample and applies a fusion rule to determine the final predicted class.Note that the pseudocode assumes the availability of functions such as train Binary SVM to train the binary SVM classifier and computeScore to compute the score or distance for a given sample. These functions can be implemented using standard SVM training algorithms and decision functions. Customizing the SVM training and decision function implementation is important based on specific requirements, such as kernel functions, regularization parameters, and optimization algorithms. This pseudocode provides a basic framework for understanding the RMSVM algorithm and can be further adapted or expanded to fit the specific needs of your implementation.

3.5. Fusion Rule (Max-Rule)

The max rule, or the maximum rule, is a fusion rule that selects the ultimate value or decision from a set of available sources. It is commonly used when the sources provide independent and complementary information, and the fusion

aims to capture the most optimistic or optimistic outcome. In decision-making or classification tasks, the max-rule can be applied to combine the outputs of multiple classifiers or models. Each classifier generates a prediction or assigns a confidence score to different classes, and the max-rule selects the class label with the highest confidence or probability among all the classifiers. This approach assumes that each classifier provides unique insights and that the highest-confidence prediction is the most reliable or accurate. The mathematical model for the max-rule fusion can be labeled as follows: Given a set of N sources, each providing a value or decision, denoted as x1, x2, ..., xN, the max-rule selects the maximum value among them. Mathematically, the max-rule fusion operation can be defined as:

$$FusedValue = max(x1, x2, \ldots, xN) \quad (13)$$

The "max" function returns the maximum value among the inputs in this equation. The max-rule is straightforward to implement and can be effective when reliable sources provide diverse information.

4. Results and Discussion

This research collected a dataset from kaggle.com for the soybean leaf disease. Dataset is a collection of images specifically focused on soybean leaf diseases. It is a widely used dataset in plant pathology for studying and progressing algorithms related to soybean disease detection and classification. The dataset comprises three image folders: caterpillar, diabrotica speciosa, and healthy. images are of soybean leaves damaged by caterpillar, diabrotica speciosa healthy images. These images were standardized by dimension 500 x 500. There are 6,410 images: caterpillar - 3,309, diabrotica speciosa - 2,205, and healthy - 896. Remove any unwanted noise or artifacts present in the images. Common noise reduction techniques include Gaussian filtering, median filtering, or morphological operations like erosion and dilation as shown in Figure 2.

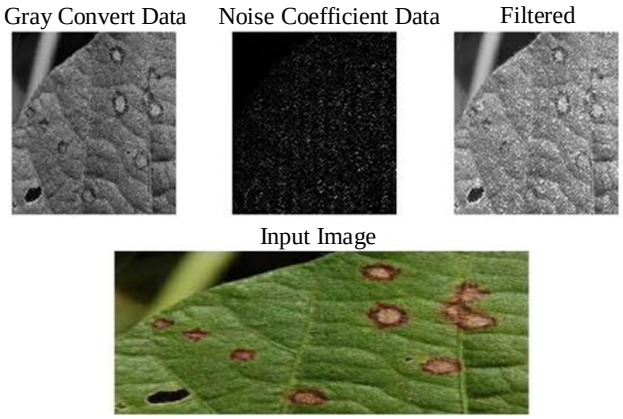

Fig. 2 Input image and preprocessed image

Texture analysis in image processing involves extracting and analyzing patterns, structures, and variations in pixel intensities within an image. The GLCM captures the statistical relationships between pairs of pixels with specific spatial relationships within an image. It computes various statistical measures such as contrast, energy, homogeneity, and entropy, which can be used as texture features, as shown in Figure 3.

Fig. 3 Texture analysis

Once the texture features are extracted, a redundant multiclass SVM model is constructed, as shown in Figure 4. Unlike traditional SVMs designed for binary classification, redundant multiclass SVM extends SVMs to handle multiple classes.

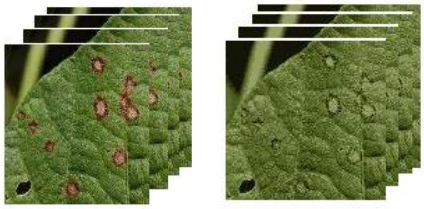

Fig. 4 Redundant multiclass SVM

Redundancy removal using K-means clustering is a technique employed to eliminate redundant or highly similar data points in a redundant multiclass SVM. Select a representative data point from each cluster to preserve the necessary information while removing redundant points, as shown in Figure 5. The Max-Rule combines multiple information or outputs from different sources or sensors to make a final decision or inference.

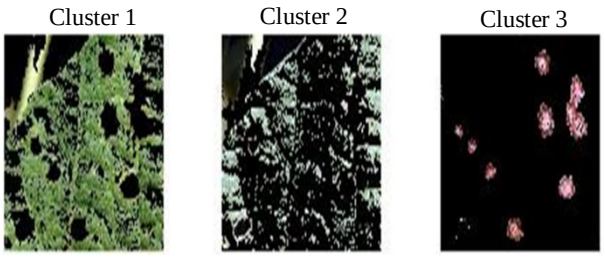

Fig. 5 Redundancy removal using K-means clustering

Fig. 6 Fusion rule to predict plant disease

The Max-Rule selects the maximum value or maximum confidence level among the available inputs, as shown in Figure 6.

The availability of the soybean leaf dataset promotes advancements in plant pathology research, improves agricultural practices, and helps mitigate crop losses due to soybean diseases. During the training phase, the redundant multiclass SVM model learns the patterns and characteristics of different leaf diseases using labeled examples. The model optimizes the decision boundaries for each class, aiming to maximize the separation between classes based on the extracted texture features, as shown in Table 1 and in Figures 7-9.

Table 1. Effectiveness of sick and typical leaf image identification

Leaf Types	Normal	Bacterial Blight	Blast	Brown Spot	Sheath rot
Accuracy	96.2	96.1	97.3	96.5	96.4
F1 Score	81.5	88.56	98.64	85	91.78
Precision	73	82.6	92.87	84	74
FPR	6.8	6	2	6.3	7
FNR	10	17	7.1	9	10
TPR	70	92.62	92.35	88.3	73.79
TNR	89	93	98	96.3	97.4
NPV	91.5	94	97	93.5	96.8

Where, FPR- False Positive Rate, FNR-False Negative Rate, TPR-True Positive Rate, TNR- True Negative Rate, NPV- Net Present Value

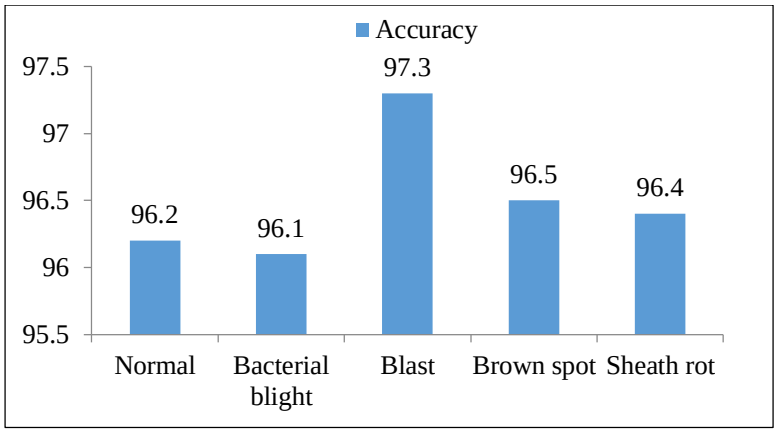

Fig. 7 Accuracy for various disease prediction

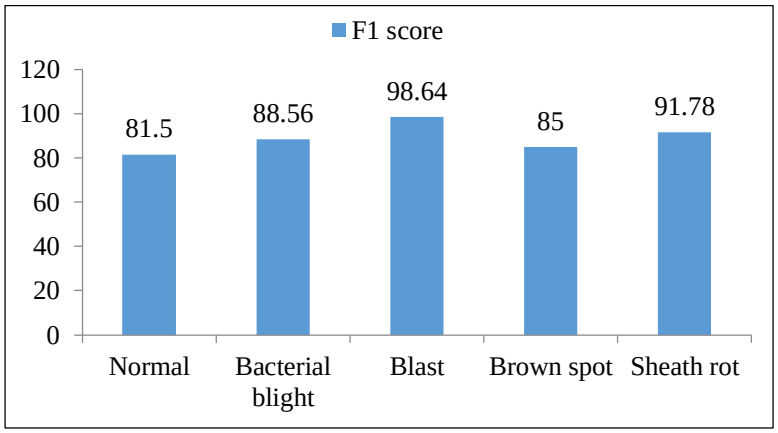

Fig. 8 F1 score for various disease prediction

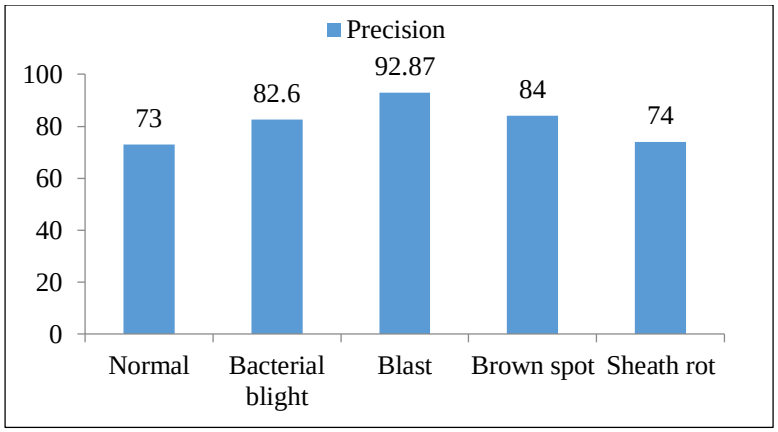

Fig. 9 Precision for various disease prediction

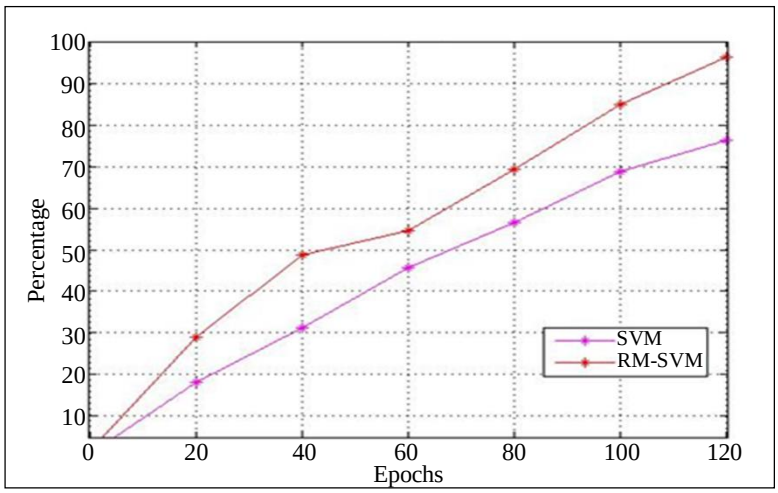

Fig. 10 Accuracy comparison

The texture features-based detection of plant leaf diseases using redundant multiclass SVM offers several advantages. It provides an automated and efficient disease detection and classification approach, allowing for rapid screening of large quantities of plant samples. By accurately identifying the presence and type of diseases, this method aids in early intervention and effective disease management, reducing crop losses and improving agricultural productivity.

The above comparison of Figure 10 shows that performance evaluation of various leaf disease predictions of RM-SVM is 95.6%, higher than existing models. To mitigate computational expenses, the approach integrates K-Means Clustering to eliminate redundancy effectively. This process is followed by applying the Max-Rule technique to fuse predicted result scores.

5. Conclusion

Our proposed model has several advantages, including non-destructive and automated disease diagnosis, which facilitates early detection and intervention. It aids in optimizing crop management practices, reducing crop losses, and enhancing agricultural productivity. Texture features-based detection of plant leaf diseases using redundant multiclass SVM is a promising approach in plant pathology. By leveraging texture analysis techniques and the power of redundant multiclass SVM, this method enables accurate and efficient detection and grouping of plant leaf diseases. The redundant multiclass SVM extends traditional SVM to handle multiple classes, improving classification accuracy to 95.6%. in plant leaf disease detection. By training various SVM classifiers, each distinguishing one class from the rest and combining their results, the redundant multiclass SVM ensures robust and reliable classification performance.

References

[1] Debangshu Chakraborty, and Indrajit Ghosh, "Image Processing Techniques in Plant Disease Diagnosis: Application Trend in Agriculture," *Information and Communication Technology for Competitive Strategies (ICTCS 2021)*, pp. 691-705, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[2] Rabbia Mahum et al., "A Novel Framework for Potato Leaf Disease Detection Using an Efficient Deep Learning Model," *Human and Ecological Risk Assessment: An International Journal*, vol. 29, no. 2, pp. 303-326, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[3] Y. Rakesh Kumar, P. Satyanarayana Goud, and Sheelam Pravalika, "Comparative Analysis on Mulberry Leaf Disease Detection Using SVM and PNN," *Cybernetics, Cognition and Machine Learning Applications*, pp. 141-146, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[4] S. Prema, and K. Radha, "Detection of Plant Leaf Diseases Using an Image," *International Journal of Early Childhood Special Education*, vol. 14, no. 4, 2022. [Google Scholar]

[5] Monu Bhagat, and Dilip Kumar, "A Comprehensive Survey on Leaf Disease Identification & Classification," *Multimedia Tools and Applications*, vol. 81, pp. 33897-33925, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[6] Asha Patil, and Kalpesh Lad, "Chili Plant Leaf Disease Detection Using SVM and KNN Classification," *Rising Threats in Expert Applications and Solutions*, pp. 223-231, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[7] Majji V. Applalanaidu, and G. Kumaravelan, "A Review of Machine Learning Approaches in Plant Leaf Disease Detection and Classification," *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, Tirunelveli, India, pp. 716-724, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[8] Rohmat Indra Borman et al., "Identification of Herbal Leaf Types Based on their Image Using First Order Feature Extraction and Multiclass SVM Algorithm," *2021 1st International Conference on Electronic and Electrical Engineering and Intelligent System (ICE3IS)*, Yogyakarta, Indonesia, pp. 12-17, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[9] Yangala Raghavendra, "Multivariant Disease Detection from Different Plant Leaves and Classification Using Multiclass Support Vector Machine," *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol. 12, no. 13, pp. 546-556, 2021. [Google Scholar] [Publisher Link]

[10] Rabia Saleem et al., "Mango Leaf Disease Recognition and Classification Using Novel Segmentation and Vein Pattern Technique," *Applied Sciences*, vol. 11, no. 24, pp. 1-12, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[11] S. Prabu et al., "Tea Plant Leaf Disease Identification Using Hybrid Filter and Support Vector Machine Classifier Technique," *Advances in Data Science and* Management, pp. 591-601, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[12] Sunil S. Harakannanavar et al., "Plant Leaf Disease Detection Using Computer Vision and Machine Learning Algorithms," *Global Transitions Proceedings*, vol. 3, no. 1, pp. 305-310, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[13] Gurubelli Yogeswararao, R. Malmathanraj, and P. Palanisamy, "An Improved Content-Based Image Retrieval System for Tomato Leaf Disease Classification," *Computational Intelligence in Machine Learning*, pp. 205-212, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[14] K.S. Archana et al., "A Novel Method to Improve Computational and Classification Performance of Rice Plant Disease Identification," *The Journal of Supercomputing*, vol. 78, pp. 8925-8945, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[15] Darah Aqel et al., "Extreme Learning Machine for Plant Diseases Classification: A Sustainable Approach for Smart Agriculture," *Cluster Computing*, vol. 25, pp. 2007-2020, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[16] Arshiya S. Ansari et al., "Improved Support Vector Machine and Image Processing Enabled Methodology for Detection and Classification of Grape Leaf Disease," *Journal of Food Quality*, vol. 2022, pp. 1-6, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[17] Vibhor Kumar Vishnoi, Krishan Kumar, and Brajesh Kumar, "A Comprehensive Study of Feature Extraction Techniques for Plant Leaf Disease Detection," *Multimedia Tools and Applications*, vol. 81, pp. 367-419, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[18] A. Ekiz, and S. Arıca, "Detection of Watermelon in RGB Images via Unmanned Aerial Vehicle by Utilising Texture Features for Predicting Yield," *Pakistan Journal of Agricultural Sciences*, vol. 59, no. 6, 2022. [Google Scholar]

[19] K.R. Bhavya et al., "An Efficient Machine Learning Approach for Apple Leaf Disease Detection," *Intelligent Computing and* Applications, pp. 419-429, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[20] Muh. Syahabuddin Hylmi, Wiharto, and Esti Suryani, "Detection of Potato Leaf Disease Using Multi-Class Support Vector Machine Based on Texture, Color, and Shape Features," *2022 International Conference on Electrical and Information Technology (IEIT)*, Malang, Indonesia, pp. 20-24, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[21] Soarov Chakraborty, Shourav Paul, and Md. Rahat-uz-Zaman, "Prediction of Apple Leaf Diseases Using Multiclass Support Vector Machine," *2021 2nd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*, Dhaka, Bangladesh, pp. 147-151, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[22] Ahmad Almadhor et al., "AI-Driven Framework for Recognition of Guava Plant Diseases through Machine Learning from DSLR Camera Sensor Based High Resolution Imagery," *Sensors*, vol. 21, no. 11, pp. 1-19, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[23] K. Rajiv et al., "Accuracy Evaluation of Plant Leaf Disease Detection and Classification Using GLCM and Multiclass SVM Classifier," *Intelligent Learning for Computer* Vision, pp. 41-54, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[24] Sk Mahmudul Hassan et al., "Identification of Plant-Leaf Diseases Using CNN and Transfer-Learning Approach," *Electronics*, vol. 10, no. 12, pp. 1-19, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[25] R. Karthickmanoj, J. Padmapriya, and T. Sasilatha, "A Novel Pixel Replacement-Based Segmentation and Double Feature Extraction Techniques for Efficient Classification of Plant Leaf Diseases," *Materials Today: Proceedings*, vol. 47, no. 9, pp. 2048-2052, 2021. [CrossRef] [Google Scholar] [Publisher Link]