Recognizing apple leaf diseases using a novel parallel real-time processing framework based on MASK RCNN and transfer learning: An application for smart agriculture

Zia ur Rehman 1 | Muhammad Attique Khan 2 | Fawad Ahmed 1 | Robertas Damaševičius 3 | Syed Rameez Naqvi 4 | Wasif Nisar 5 | Kashif Javed 6

Abstract
Effective recognition of fruit leaf diseases has a substantial impact on agro-based economies. Several fruit diseases exist that badly impact the yield and quality of fruits. A naked-eye inspection of an infected region is a difficult and tedious process; therefore, it is required to have an automated system for accurate recognition of the disease. It is widely understood that low contrast images affect identification and classification accuracy. Here a parallel framework for real-time apple leaf disease identification and classification is proposed. Initially, a hybrid contrast stretching method to increase the visual impact of an image is proposed and then the MASK RCNN is configured to detect the infected regions. In parallel, the enhanced images are utilized for training a pre-trained CNN model for features extraction. The Kapur’s entropy along MSVM (EaMSVM) approach-based selection method is developed to select strong features for the final classification. The Plant Village dataset is employed for the experimental process and achieve the best accuracy of 96.6% on the ensemble subspace discriminant analysis (ESDA) classifier. A comparison with the previous techniques illustrates the superiority of the proposed framework.

1 INTRODUCTION
In computer vision, early effective identification of infected fruit leaves is an important challenge, mainly due to their significant utilization in agriculture in the agro-based economies [1]. Several fruit diseases exist that badly impact the yield and quality of fruits. The symptoms appearing on the leaves give information, which helps in the identification of diseases in various kinds of fruit plants [2, 3]. Detection of the infected leaves at an early stage effectively enhances the fruit quality and yield [4, 5]. Different types of diseases are identified based on the shape, colour, and texture of lesion spots, so the shape, colour, and texture of lesions are the essential features for the classification of apple diseases [6]. It has been reported that attacks of different diseases like scab, rust, and black rot severely impact the fruit crop yield and quality [7, 8]. It is critical to develop an automated computerized system to detect and classify the infected apple leaves at an early stage. Many computing specialists proposed many algorithms for the computerized/automated identification of plant and fruit diseases [9–11].

Several techniques like image segmentation [12], feature extraction [13], and classification are presented [14–16]. Data augmentation and transfer learning techniques are used for the classification of images using deep learning [17, 18]. Particularly for the fruit disease segmentation and recognition, various feature-based combinations are proposed, including correlation-based feature selection, colour-based and weighted segmentation, texture features, shape features, support vector machine (SVM), and some others [19]. However, detection of the leaf disease remains a difficult task because of the following factors: i) on the same leaf, multiple lesions may occur; ii) size of the lesion may vary on the same leaf; iii) the lesion spots...
of apple leaf diseases are tiny; and iv) interference of ambient factors such as illumination and shadow affects the detection of lesion spots on leaves [20].

Recent computational developments, particularly embedded graphic processing units (GPUs), have revolutionized machine learning; as a result of which, novel advanced models and techniques have been proposed [21]. Deep learning is one of those techniques, which relates to the adoption of artificial neural network (ANN) architectures with many processing layers [22]. The deep learning approach has shown huge success in several computer vision applications like medical imaging, assisted living, and object classification [23–25]. However, in the agriculture domain, deep learning is mostly employed for classification purposes [26, 27]. However, the segmentation of the tiny infected regions is not an easy process due to the presence of the ground truth images [28].

Several researchers have introduced computer vision-based techniques in the literature. Dubey et al. [29] use several features to describe a new computer vision-based technique. The technique comprises K-means segment three main steps: i) lesion spots; ii) features extraction like colour, LBP, and shape from the segmented spots, and iii) application of the multiclass SVM (MSVM) classifier for the final classification. Experiments are performed on several apple diseases and achieved notable performance. Khan et al. [2] present a system for several fruit diseases using deep learning. This technique describes two main steps—disease segmentation using correlation-based strategy and disease classification using deep features. Ormani et al. [30] present a computerized technique for apple disease identification. Three main steps are involved in the presented method: i) K-means clustering is employed for segmentation of lesion spots; ii) GLCM, colour, and wavelet features are extracted from the segmented spots, and iii) extracted features are fed in support vector regression classifier for the classification. Shuaibu et al. [31] introduce an optimized stochastic algorithm for the detection of the apple disease named Marssonina.

Optimal features are chosen by using the particle swarm optimization (PSO) algorithm based on the highest discrimination. Then the SVM classifier does the classification and manages to achieve improved accuracy. Barbedo et al. [32] present a methodology by addressing colour channel and a Boolean operation to segmentation lesion spots on infected plant leaves. This methodology applies a binary mask. The challenges like variability in illumination and environment are resolved by this method.

In this work, we propose a framework for apple leaf disease detection and classification. For this purpose, we generate ground truths to train a CNN model name MASK RCNN. Our main contributions in this work are as follows:

- A hybrid contrast stretching technique is implemented based on the fusion of two filters and then increase the intensity values by replacing a threshold value.
- A MASK RCNN is trained by backbone Resnet101+FPN to detect the infected regions.
- Deep learning features are extracted using a pre-trained CNN model. The transfer learning-based features are extracted, and the best of them are selected using the proposed Kapur entropy with MSVM (EaMSVM) approach.

• We compare the performance of average pool layer features and fully connected layer features based on the accuracy and testing time.

2 | PROPOSED METHOD

This section presents our proposed scheme using deep learning for apple fruit disease detection and classification. Figure 1 shows the proposed model, which consists of two parallel steps: disease detection and disease classification. A hybrid contrast enhancement method is implemented at the very first to improve the quality of images. Later, the improved images are passed onto MASK RCNN as well as re-training of ResNet-50 for classification. The detail of each step, shown in this figure, is as follows:

2.1 | Hybrid contrast stretching

Contrast stretching is an important step in computerized methods that deal with the identification of specific regions or objects, such as infected parts [33]. High-quality images usually prove helpful in extracting the infected region in an image with good precision rates. Here we implement a hybrid contrast stretching approach to increase the visual quality of an image. The presented approach is based on the fusion of top-bottom filters, and then set a mean value based threshold function. Based on the latter, the intensity of pixel values is increased. This approach is simple but most effective for this work. Mathematically, this method is presented as follows:

Consider an input image of dimension $256 \times 256 \times 3$, then a top-hat filter is defined as:

$$
\text{top}(i, j) = \psi(I, s)
$$

(1)

where $\psi$ is the top-hat activation function, $I$ is an original image, and $s$ is the structuring element which is initialized as 11 in this work. The bottom hat filter is defined as follows:

$$
\text{bot}(i, j) = \Psi(I, s)
$$

(2)

where $\Psi$ denotes the bottom hat activation function and $s$ is the structuring element and its value is 5 for this work. After that, we add the information of top$(i, j)$ along with the original image as:

$$
\text{Ne}(i, j) = \left( \text{top}(i, j) + I \right)
$$

(3)

The new enhanced image New$(i, j)$ is improved by the contrast of local regions, which is further improved by the addition of the bottom hat image as:

$$
\text{New}(i, j) = \left( \text{Ne}(i, j) + \text{bot}(i, j) \right)
$$

(4)
The mean for this new image is calculated and put into a threshold function, which is defined as:

\[
T = \begin{cases} 
    \text{New}(i,j) & \text{if New}(i,j) < \mu \\
    \text{otherwise, do not update} & 
\end{cases}
\]  

(5)

The New\(_i\)\(_j\) is the final contrast-enhanced image. The resultant image is further passed into MASK RCNN for the disease spot identification.

### 2.2 Modified mask RCNN

Segmentation of the image is used to identify different changes in plant leaves. The real challenge is to identify the precise pixel boundaries of the infected area. The precise location of the infected area helps farmers to take timely targeted counter actions to avoid further spread of the infection. In this work, Mask RCNN [34] is utilized to segmentation infected regions using contrast-enhanced images. Mask RCNN is an evolved version of the faster RCNN, and it is more efficient than the faster RCNN. Mask RCNN consists of several parts, which aid each other in getting the desired output. These parts, named backbone, region proposal network (RPN), ROI alignment, network head, and loss function, are illustrated in Figure 2.

**Backbone:** This is the standard CNN (usually, ResNet) that acts as the extractor of features. The low-level features are extracted from the early layers while the high-level ones are extracted from the latter layers. The convolutional neural network backbone creates feature maps as input image passes through it. Consider, we have a contrast stretching image of dimension 256 \( \times \) 256 \( \times \) 3. As the acceptable input of Mask RCNN is dimension 1024 \( \times \) 1024 \( \times \) 3, so we converted enhanced images into a required input size of a network. The images are resized using a simple function of the programming tool Matlab (imresize). Each input image is converted from 1024 \( \times \) 1024 \( \times \) 3 to feature map size 32 \( \times \) 32 \( \times \) 2048. This feature map acts as input to further layers of the network. Feature pyramid network (FPN) [35] is employed to aid ResNet during the extraction of features. Using FPN, objects can be represented better at multiple scales. FPN facilitates both low-level and high-level features to be accessed at every level. We use separate combinations for the implementation of Mask RCNN: 1) ResNet50+FPN backbone and 2) Resnet101+FPN backbone.

Regional proposal network (RPN) proposes a potential candidate for object bounding boxes by utilizing the image feature of the backbone. This network is a replacement of the selective search [36], which is a computationally slower mechanism. The scanning is performed by RPN over the regions of the image are called anchors. They are overlapped and spread all over the image to find the potential candidate for bounding boxes in different sizes and aspect ratios. Anchors having the highest probability of RPN proposals are chosen for further refinement. As anchors overlap, so to reduce redundancy among the RPN proposals, non-maximum suppression [37] are used based on their class scores. The RPN loss function is described as:

\[
\lambda_i (b_i, B_i) = \frac{1}{N_c} \sum_j L_c (b_i, b_j^*) + \psi \frac{1}{N_{\text{reg}}} \sum_j b_i^* I_{\text{reg}} (B_i, B_i^*)
\]  

(6)

Here the predicted probability of an anchor point \( i \), is represented by \( b_i \). Ground truth label is denoted by \( b_i^* \). It is represented by 1 if the anchor point is positive otherwise, 0.

The bounding box-segmented area is drawn based on the four coordinates provided by the vector \( B_i \) and \( B_i^* \) represent the ground truth bounding box, respectively.
ROI align: Mask RCNN requires the ROI features to be positioned to keep the spatial correspondence per pixel accurately to predict pixel masks. For this reason, the ROI align layer in Mask R-CNN replaced the ROI Pool layer in Faster R-CNN. For the computation of the precise location of input features, the ROI align layer uses bi-linear interpolation [38] at four regularly sampled locations in every bin of ROI. For further refinement, average or max pooling is carried out on already attained features.

Network Head: Network head is a fully connected layer used to perform classification, prediction of the segmented mask, and bounding box generation. The input of the network head are features extracted from ROI align. The network head performs these tasks simultaneously. Against each ROI, a mask of size \( m \times m \) is predicted by the fully connected layer. It is necessary to preserve the definite dimensional layout of \( m \times m \) object.

The utilized loss function is defined as follows:

\[
\lambda = \lambda_{\text{class}} + \lambda_{\text{box}} + \lambda_{\text{mask}}
\]

The detail of parameters initialized for the configuration of MASK RCNN, is given in Table 1.

Table 1 shows that per epoch training steps are 400 and total validation steps are 50. Moreover, the learning rate is 0.01, and the momentum is 0.6. The resultant segmented images are shown in Figure 3, where the second column shows the ground truth images, the third column shows the bounding box images, and the last column represents the final segmented images, respectively.

### Classification task

Today computerized methods are used for classification problems. In the field of agriculture, these computerized methods are also applied. These methods aid agriculture experts to classify multiple diseases. If we talk about agriculture, then deep convolutional neural networks outperform the classical methods for the classification of complex agriculture diseases in terms of both computational time and accuracy. However, the main problem is the availability of large datasets, because a deep learning model requires a handsome amount of data for training.

#### 2.4 Feature extraction using TL

A pre-trained ResNet-50 model has been trained on selected fruits images through transfer learning (TL). TL is the process of reuse a pre-trained model for a new task with less computational time. A complete process of TL is shown in Figure 4. As mentioned in Figure 1, the Resnet50 pre-trained convolutional neural network is employed and retrained on contrast-enhanced apple leaf images. Originally, the ResNet model [39] is trained on the ImageNet dataset [40]. This model is composed of total 177 layers, which include 53 of the convolutional layers, 49 of the ReLu layers, 53 of the batch normalization layers, and 16 additional layers. The architecture of ResNet50 is given in Table 2. This model is fine-tuned for the new task. In the fine-tuning, the last layer has been removed and added a new layer for which includes only four types of apple diseases as mentioned in the results section. After that, employed an activation function on the feature layers (Avg Pool, FC) and obtained two feature vectors. During the training of a deep learning model, we used the following hyper parameters such as Mini batch size of 64, Momentum is 0.93, learning rate is 0.0001, and weight decay 0.2. Visually, the TL concept is shown in Figure 4.
In our work, we use cross-entropy as the activation function. The following expression defines the cross-entropy activation function:

$$H_c(f_i, C) = - \sum_{m}^{M} P_{i,c} \log (P_{i,c})$$  (8)

here, $M$ is the number of classes, $C$ are the class labels, and $P$ is the probability for $N$ observations over the class $m$.

To get the minimum value of the loss function, optimization algorithms are used. Here gradient descent is used as the optimization algorithm. Choosing a good learning rate is crucial to achieving the local minima. It defines how fast or slow our system converges. With a well-chosen learning rate, the system attains better accuracies, as well as it becomes time efficient.

Mathematically, gradient descent is defined as

$$\theta = \theta - \eta \times \nabla_{\theta} J(\theta)$$  (9)

Mini batch gradient descent is one variant of gradient descent. It learns faster and more frequently as it goes through a lesser number of examples per batch. It avoids redundant examples as it selects random examples. One more advantage of the mini-batch gradient is that it improves the generalization error, as the batch size is smaller than the training dataset. Mathematical equations for the mini-batch gradient descent is defined as follows:

$$\theta = \theta - \eta \times \nabla_{\theta} J(\theta, x^{(i+1)}, y^{(i+1)})$$  (10)

where $\eta$ is the learning rate, $x^{(i)}$ and $y^{(i)}$ are the training examples, and $n$ is the size of the mini-batch. Momentum and learning rate are 0.93 and 0.001, respectively. After applying the activation function on two layers named average pool layer and fully connected layers, we obtained two feature vectors of dimensions $N \times 2048$ and $N \times 1000$, respectively. Later, we select the best features from both of them using the Kapur entropy along with the SVM approach and perform classification.

### 2.5 Feature selection and classification

The extraction of specific features from the original extract feature space is important to improve accuracy and minimum computational time. Several feature selection techniques are presented in the literature for the selection of the most discriminant feature points. The main purpose of feature selection in this work is to reduce the feature-length and minimize the execution time during the testing process. For this purpose, we propose using a Kapur entropy along with SVM (EaSVM) approach. In this approach, initially, the Kapur entropy is calculated. Mathematically, Kapur entropy is defined as follows:

$$E_1 = - \sum_{j=1}^{h_0} \frac{p_j}{h_0} \ln \frac{p_j}{h_0}, \quad h_0 = \sum_{j=1}^{h} p_j$$  (11)

$$E_n = - \sum_{j=1}^{l-1} \frac{p_j}{h_n} \ln \frac{p_j}{h_n}, \quad h_n = \sum_{j=1}^{l-1} p_j$$  (12)

where, $E_n$ denotes entropies of the apple leaf classes and $p_j$ denotes probability values of each class. Based on $E_n$, we define an activation function to maximize the entropy values: $E_{mx} = \arg\max \{E_n\}$. The value of $E_{mx}$ is embedded in a threshold function, and the features, equal to or greater than $E_{mx}$ are selected, and represented by $E_d$. The selected features based on this threshold are embedded in the Multiclass SVM. The MSVM is employed as fitness functions and computes the error rate. This process is continued up to 50 times, and the best feature vector based on the minimum error rate is selected. After the selection of the best feature vector, several classification
methods are employed, and the best classifier based on high accuracy is selected.

3 | EXPERIMENTAL RESULTS AND DISCUSSION

This method is validated on the Plant village dataset [41]. The proposed method results are divided into two parts: i) Segmentation results of the implemented contrast enhancement and MASK RCNN, ii) Classification results of apple diseases using deep learning.

3.1 | Experimental setup

A few sample images from the plant village dataset are shown in Figure 5. VGG image annotator (VIA) is used for manual annotation of the dataset. Few samples of the manually annotated images are shown in Figure 6. The infected leaf regions are separated by making a mask. These annotated images are saved in the form of a JSON file. For training, we used that JSON file as ground truths. Apple scab, Black rot, and cedar rust images are used for segmentation. In the evaluation of the segmentation task, an 80:20 is employed. As we have a total of 1200 ground truth images, we use 240 images for testing. In the classification task, a ratio of 70 to 30 is selected for training and testing. Further details about the dataset like per class images, total no of images, and the number of training and testing images are presented in Table 3.

The Resnet50 pre-trained model is utilized for deep features extracted from the last two layers—average pooling and fully connected. A cross-validation strategy is used to compute classification results over different K folds such as $K = 5, 10, 15, 20$. Ten different classification methods are used to evaluate the validation of the proposed system. The classification methods include linear discriminate, SVM linear, SVM quadratic, SVM cubic, medium Gaussian SVM, coarse Gaussian, medium KNN, cubic KNN, ensemble bagged tree, ensemble subspace discriminant. The performance of classification is evaluated on the number of parameters including sensitivity, specificity, precision AUC, FPR, and accuracy. Execution time is also calculated to see which method is more time-efficient. Classification results are obtained by using Core i7 4770 CPU, 16 Gb RAM, and NVIDIA GTX 1070 GPU.

3.2 | Result analysis of segmentation using ResNet

The evaluation of the segmentation model is done by calculating precision and recall values at multiple threshold values of intersection over union (IOU). The considered threshold values are 0.5, 0.75, and 0.85 for segmentation results. Precision and recall values at these threshold values are calculated as given in Tables 4 and 5 (Resnet50+FPN). With the increase in the threshold value, we note that the precision and recall values start to decline. At a lower threshold value, the false rate is maximized while the higher values of the threshold minimize such false identification. As compared to training network heads only, we obtain better results on training all the layers of the network.
A similar pattern is observed when we perform a similar experiment for backbone Resnet101+FPN. The results are presented in Tables 6 and 7. By training network heads only with all layers, the calculated precision is 0.793 and recall is 0.532 and 0.554, respectively for 0.5-threshold value. After obtaining the results for both backbones, we can infer that the results of the backbone Resnet101 are better.

Average precision (AP) is computed for better evaluation of each case at different values of threshold, which is 0.5, 0.75, and 0.85. The computed average precision is shown in Tables 8 and 9. The highest value of AP is observed when all the layers of the network are trained by using Resnet101 as the backbone. We attain an AP value of 0.861 at the threshold value of 0.5. From the results, it is observed that training all the layers of the model with Resnet101+FPN (R101+FPN) as a backbone gives better performance.

### 3.3 Result analysis for classification

In this section, the classification results are presented. Table 10 shows the classification results on the average pooling layer for

| BACKBONE | AP 0.5 | AP 0.75 | AP 0.85 | MoC |
|----------|--------|--------|--------|-----|
| R50      | 0.836  | 0.479  | 0.138  | 0.229 |
| R101+FPN | 0.861  | 0.489  | 0.153  | 0.225 |

**TABLE 5** Segmentation results on training all layers

| Threshold | Precision | Recall |
|-----------|-----------|--------|
| 0.5       | 0.797     | 0.540  |
| 0.75      | 0.593     | 0.417  |
| 0.85      | 0.197     | 0.249  |

**TABLE 6** Segmentation results on backbone Resnet101+FPN for network heads only

| Threshold | Precision | Recall |
|-----------|-----------|--------|
| 0.5       | 0.793     | 0.532  |
| 0.75      | 0.505     | 0.374  |
| 0.85      | 0.103     | 0.215  |

**TABLE 7** Segmentation results on backbone Resnet101+FPN for training all layers

| Threshold | Precision | Recall |
|-----------|-----------|--------|
| 0.5       | 0.793     | 0.554  |
| 0.75      | 0.583     | 0.420  |
| 0.85      | 0.207     | 0.248  |

**TABLE 8** Training on network heads, where MoC represents mean overlapping co-efficient

| BACKBONE   | AP 0.5 | AP 0.75 | AP 0.85 | MoC   |
|------------|--------|--------|--------|------|
| R50+FPN    | 0.774  | 0.369  | 0.062  | 0.219|
| R101+FPN   | 0.818  | 0.374  | 0.069  | 0.223|

different types of classifiers, and each classifier is validated by cross-validation. The best performance attained on linear discriminate (LD) when compared to other classifiers using the average pooling layer features. The attained accuracy is 99.1% along with the sensitivity is 99.25%, specificity is 99%, precision is 99.25%, area under the curve is 0.9925, FPR is 0.0025 where $K = 5$. This accuracy of LD is also validated through a confusion matrix in Figure 7.

In Figure 7a, it is clearly shown that the proposed method on $K = 5$ gives better performance. The class apple rust gives 100% accuracy whereas the other three classes reach 99%. The second-best accuracy is attained on LD, which is 98.8%, while sensitivity, specificity, precision, AUC, and FPR are 99%, 99%, 98.75%, 0.9925, and 0.0025, respectively over 10-folds cross-validation. The confusion matrix which is shown in Figure 7 also validates these results.

Figure 7b shows that apple rust gives 100% accuracy, apple scab, and black rot reach 99% whereas, the class healthy apple attains 98% accuracy. For $K = 15$, the best-attained accuracy is 98.8% on LD along with the sensitivity of 98.75%, the specificity of 96%, precision of 98.75%, AUC of 0.99, and FPR of 0.005. Figure 7c shows that apple scab and apple rust attain accuracy of 100%, whereas black rot and healthy class give 99% and 96% accuracy, respectively. On $K = 20$, the best classification accuracy is attained by LD, which is 99.1% whereas sensitivity is 99.25%, specificity is 98%, precision is 99%, AUC is 0.9925, and FPR is 0.0025. Processing time is also an important parameter for a real-time system, so we compute processing time for each classifier, as given in Table 10. The time consumption of each classifier using the average pooling layer is shown in Figure 8.

Table 11 shows the results of the proposed classification method on the fully connected layer. The best performance attained on ensemble subspace discriminate (ESD) when compared to other classifiers using the fully connected layer features of $K = 5$. The attained accuracy is 96.6% along with the sensitivity rate of 96.5%, specificity is 96%, precision is 96.75%, area under the curve is 1, and FPR is 0.01.

This accuracy of ESD is also validated through a confusion matrix in Figure 9. Figure 9a shows that the proposed method on $K = 5$ gives better performance. The class apple rust gives 100% accuracy whereas black rot and healthy give 96% and scab reached 95%. When the classification accuracy is computed on $K = 10$, we get the best accuracy using the CSVM classifier of 98.2% along with a sensitivity rate of 98.5%, specificity is 98%, precision is 98%, AUC is 0.9925, and FPR is 0.005.

Figure 9b shows that the class apple scab and black rot give 99% accuracy, while others give 98% accuracy. For $K = 15$, the QSVM classifier attains the best accuracy of 98.5% which is the highest accuracy among all the classifiers. The other
| Method                          | K-folds | Performance measures |
|--------------------------------|---------|-----------------------|
|                                | K = 5   | K = 10 | K = 15 | K = 20|
| Linear discriminate            | ✓       | ✓      | ✓      | ✓     |
|                               | Sensitivity (%) | Specificity (%) | Precision (%) | AUC | FPR | Accuracy (%) | Time (s) |
| Linear discriminate            | 99.25   | 99     | 99.25  | 0.9925 | 0.0025 | 99.1 | 8.9457    |
| SVM linear                     | ✓       | ✓      | ✓      | ✓     |
|                               | Sensitivity (%) | Specificity (%) | Precision (%) | AUC | FPR | Accuracy (%) | Time (s) |
| SVM linear                     | 99.00   | 99     | 99.00  | 1     | 0.005  | 98.8 | 6.6460    |
| SVM quadratic                  | ✓       | ✓      | ✓      | ✓     |
|                               | Sensitivity (%) | Specificity (%) | Precision (%) | AUC | FPR | Accuracy (%) | Time (s) |
| SVM quadratic                  | 97.50   | 94     | 97.25  | 1     | 0.010  | 97.3 | 12.925    |
| SVM cubic                      | ✓       | ✓      | ✓      | ✓     |
|                               | Sensitivity (%) | Specificity (%) | Precision (%) | AUC | FPR | Accuracy (%) | Time (s) |
| SVM cubic                      | 97.50   | 94     | 97.50  | 1     | 0.0075 | 97.6 | 30.30     |
| Medium Gaussian SVM            | ✓       | ✓      | ✓      | ✓     |
|                               | Sensitivity (%) | Specificity (%) | Precision (%) | AUC | FPR | Accuracy (%) | Time (s) |
| Medium Gaussian SVM            | 93.25   | 90     | 94.50  | 0.9975 | 0.0225 | 93.3 | 17.131    |
| Coarse Gaussian SVM            | ✓       | ✓      | ✓      | ✓     |
|                               | Sensitivity (%) | Specificity (%) | Precision (%) | AUC | FPR | Accuracy (%) | Time (s) |
| Coarse Gaussian SVM            | 93.50   | 89     | 93.75  | 0.995  | 0.015  | 93.6 | 23.203    |
| Medium KNN                     | ✓       | ✓      | ✓      | ✓     |
|                               | Sensitivity (%) | Specificity (%) | Precision (%) | AUC | FPR | Accuracy (%) | Time (s) |
| Medium KNN                     | 88.75   | 89     | 89.25  | 0.9875 | 0.035  | 88.7 | 17.055    |
| Cubic KNN                      | ✓       | ✓      | ✓      | ✓     |
|                               | Sensitivity (%) | Specificity (%) | Precision (%) | AUC | FPR | Accuracy (%) | Time (s) |
| Cubic KNN                      | 85.50   | 83     | 86.50  | 0.9875 | 0.0475 | 85.4 | 49.172    |
| Ensemble bagged trees          | ✓       | ✓      | ✓      | ✓     |
|                               | Sensitivity (%) | Specificity (%) | Precision (%) | AUC | FPR | Accuracy (%) | Time (s) |
| Ensemble bagged trees          | 93.00   | 93     | 93.25  | 0.9925 | 0.0225 | 93   | 27.347    |
| Ensemble subspace Discriminant | ✓       | ✓      | ✓      | ✓     |
|                               | Sensitivity (%) | Specificity (%) | Precision (%) | AUC | FPR | Accuracy (%) | Time (s) |
| Ensemble subspace Discriminant | 98.25   | 96     | 98.25  | 1     | 0.005  | 98.2 | 36.274    |
TABLE 11  Proposed classification accuracy using deep features on full connected CNN layer

| Method                | K-Folds K = 5 | K = 10 | K = 15 | K = 20 | K-Folds Sensitivity (%) | Specificity (%) | Precision (%) | AUC     | FPR     | Accuracy (%) | Time (s) |
|-----------------------|----------------|--------|--------|--------|--------------------------|----------------|--------------|---------|---------|-------------|----------|
| Linear discriminate   | ✓              | ✓      | ✓      |        | 93.75                    | 93              | 94.33        | 0.9575  | 0.0225  | 93.6        | 7.397    |
|                       |                |        |        |        | 97.50                    | 98              | 97.50        | 0.980   | 0.010   | 97.3        | 7.246    |
|                       |                |        |        |        | 96.00                    | 95              | 96.25        | 0.9725  | 0.010   | 96.0        | 5.654    |
| SVM linear            | ✓              | ✓      | ✓      | ✓      | 94.75                    | 90              | 94.50        | 0.965   | 0.0175  | 94.0        | 5.5881   |
|                       |                |        |        |        | 97.00                    | 96              | 96.75        | 1       | 0.010   | 97.0        | 13.852   |
| SVM quadratic         | ✓              |       | ✓      | ✓      | 97.25                    | 96              | 96.50        | 0.9975  | 0.010   | 96.3        | 11.21    |
| SVM cubic             | ✓              |       | ✓      | ✓      | 98.25                    | 98              | 97.75        | 1       | 0.005   | 97.9        | 15.174   |
|                       |                |        |        |        | 98.75                    | 98              | 98.75        | 1       | 0.005   | 98.5        | 16.725   |
| Medium Gaussian SVM   | ✓              | ✓      | ✓      | ✓      | 95.25                    | 91              | 94.25        | 0.9975  | 0.0175  | 93.7        | 13.205   |
|                       |                |        |        |        | 96.25                    | 96              | 96.25        | 1       | 0.010   | 96.3        | 18.608   |
| Coarse Gaussian SVM   | ✓              |        | ✓      | ✓      | 93.75                    | 91              | 94.25        | 0.9975  | 0.0175  | 93.7        | 23.203   |
| Medium KNN            | ✓              |       | ✓      | ✓      | 91.75                    | 89              | 91.75        | 0.9875  | 0.0275  | 91.8        | 13.44    |
|                       |                |        |        |        | 89.50                    | 85              | 90.00        | 0.9875  | 0.035   | 89.3        | 20       |
| Cubic KNN             | ✓              |       | ✓      | ✓      | 89.00                    | 87              | 89.75        | 0.9875  | 0.0375  | 89.0        | 21.788   |
|                       |                |        |        |        | 89.50                    | 85              | 89.50        | 0.990   | 0.0325  | 89.5        | 21.276   |
| Ensemble bagged trees | ✓              |       | ✓      | ✓      | 91.25                    | 87              | 91.25        | 0.9875  | 0.0275  | 91.2        | 27.45    |
|                       |                |        |        |        | 89.00                    | 82              | 89.50        | 0.9875  | 0.0375  | 89.0        | 35.97    |
| Ensemble subspace discriminant | ✓ |       | ✓      | ✓      | 88.50                    | 84              | 89.00        | 0.9875  | 0.0375  | 88.4        | 40.105   |
|                       |                |        |        |        | 88.00                    | 82              | 88.00        | 0.990   | 0.040   | 87.5        | 45.999   |
|                       | ✓              |       | ✓      | ✓      | 91.50                    | 89              | 91.25        | 0.9875  | 0.0275  | 91.5        | 22.524   |
|                       |                |        |        |        | 89.50                    | 85              | 89.50        | 0.990   | 0.0325  | 89.5        | 35.25    |
|                       | ✓              |       | ✓      | ✓      | 92.75                    | 95              | 97.50        | 1       | 0.005   | 97.6        | 63.002   |
|                       |                |        |        |        | 89.00                    | 84              | 89.00        | 0.985   | 0.0375  | 89.0        | 43.385   |
|                       | ✓              |       | ✓      | ✓      | 96.50                    | 96              | 96.75        | 1       | 0.010   | 96.6        | 28.837   |
|                       |                |        |        |        | 97.50                    | 95              | 97.50        | 1       | 0.0075  | 97.6        | 19.975   |
|                       | ✓              |       | ✓      | ✓      | 92.75                    | 90              | 93.25        | 0.9925  | 0.020   | 93.0        | 43.99    |
|                       |                |        |        |        | 97.25                    | 96              | 97.25        | 1       | 0.010   | 97.3        | 63.369   |
parameters such as sensitivity rate of 98.75%, specificity are 98%, precision rate is 98.75%, AUC is 1.00, and FPR is 0.005, respectively. A confusion matrix validates these results as illustrated in Figure 9c. For $K = 20$, QSVM gives the best accuracy of 97.9%. Figure 9d shows that class apple rust gives 100% accuracy, whereas apple scab, black rot, and healthy classes give 99%, 98%, and 95% accuracy, respectively. Overall, ESD and QSVM give better results. Processing time is also computed for the fully connected layer, plotted in Figure 10.

While comparing the results of the average pooling layer and the fully-connected layer we can infer that we get better performance on the average pooling layer as compared to the latter based on accuracy and time. The average pooling layer takes minimum time to attain better accuracy. Also, a comparison with known methods is presented in Table 12, which supports

| Reference | Year | Accuracy |
|-----------|------|----------|
| [42]      | 2013 | 90%      |
| [43]      | 2015 | 94.9%    |
| [44]      | 2016 | 94.74%   |
| [45]      | 2016 | 82.3%    |
| [6]       | 2017 | 90.0%    |
| [46]      | 2017 | 90.4%    |
| Proposed  | 2017 | 99.1%    |
the proposed method’s performance. Moreover, a few labelled results of the proposed technique are shown in Figure 11.

4 CONCLUSION

A new method is presented for real-time apple leaf disease detection and classification using MASK RCNN and Deep Learning feature selection. The experimental results demonstrate the efficiency of the proposed technique for detecting and classification of apple leaf diseases. The results support our conclusion that the contrast stretching approach’s implantation influences the detection of the infected leaf regions and gives a better mean overlapping coefficient (MoC). Also, this step is useful in detecting minor spots with less error rate. For the classification step, selecting the best features minimizes the error rate and increases the precision value. Also, based on the comparison among two-layer features, it is observed that the average pool layer features are more discriminant as compared to the FC layer features.

In future studies, we will try to include a larger number of ground truth images to train the MASK RCNN for improvement of the MoC value. Also, we will consider more different kinds of fruits for the classification task.

REFERENCES

1. Savary, S., et al.: Crop Losses Due to Diseases and Their Implications for Global Food Production Losses and Food Security. Springer, Berlin (2012)

2. Khan, M.A., et al.: CCDF: Automatic system for segmentation and recognition of fruit crops diseases based on correlation coefficient and deep CNN features. Comput. Electron. Agric. 155, 229–236, (2018)

3. Oyewola, D.O., et al.: Detecting cassava mosaic disease using a deep residual convolutional neural network with distinct block processing. PeerJ Computer Science, 7(2), e352 (2020)

4. Sharif, M., et al.: Detection and classification of citrus diseases in agriculture based on optimized weighted segmentation and feature selection. Comput. Electron. Agric. 150, 220–234, (2018)

5. Iqbal, Z., et al.: An automated detection and classification of citrus plant diseases using image processing techniques: A review. Comput. Electron. Agric. 153, 12–32, (2018)

6. Chauanle, Z., et al.: Apple leaf disease identification using genetic algorithm and correlation based feature selection method. International Journal of Agricultural and Biol. Eng. 10, 74–83, (2017)

7. Khan, M.A., et al.: An optimized method for segmentation and classification of apple diseases based on strong correlation and genetic algorithm based feature selection. IEEE Access 7, 46261–46277, (2019)

8. Rauf, H.T., et al.: A citrus fruits and leaves dataset for detection and classification of citrus diseases through machine learning. Data in Brief 26, 104340, (2019)

9. Adeel, A., et al.: Diagnosis and recognition of grape leaf diseases: An automated system based on a novel saliency approach and canonical correlation analysis based multiple features fusion. Sustainable Computing: Informatics and Systems 24, 100349, (2019)

10. Capizzi, G., et al.: A novel neural networks-based texture image processing algorithm for orange defects classification. International Journal of Computer Science and Applications 13(2), 45–60, (2016)

11. Woiznak, M., Polap, D.: Adaptive neuro-heuristic hybrid model for fruit peel defects detection. Neural Networks 98, 16–33, (2018)

12. Ahmad, J., et al.: Visual features based boosted classification of weeds for real-time selective herbicide sprayer systems. Comput. Ind. 98, 23–33, (2018)

13. Tahir, M.B., et al.: Recognition of apple leaf diseases using deep learning and variances-controlled features reduction. Microprocess. Microsyst. 10:4027 (2021)

14. Auranzeb, K., Akmal, F., Khan, M.A., Sharif, M., Javed, M.Y.: Advanced Machine Learning Algorithm Based System for Crops Leaf Diseases Recognition. In: 2020 6th Conference on Data Science and Machine Learning Applications, pp. 146–151. IEEE, Piscataway, NJ (2020)

15. Khan, M.A., et al.: An automated system for cucumber leaf disease spot detection and classification using improved saliency method and deep features selection. Multimedia Tools and Applications, 79, 18627–18656 (2020)

16. Gabryel, M., Damašėvičius, R.: The image classification with different types of image features. Lect. Notes Comput. Sci. 10245, 497–506 (2017)

17. Urbonas, A., Raudonis, V., Maskelėtė, R., Damašėvičius, R.: Automated identification of wood veneer surface defects using faster region-based convolutional neural network with data augmentation and transfer learning. Appl. Sci. 9, 4898 (2019)

18. Abayomi-Alli, O.O., Damašėvičius, R., Misra, S., Maskelėtė, R.: Cassava disease recognition from low quality images using enhanced data augmentation model and deep learning. Expert Systems (2021)

19. Adeel, A., et al.: Entropy-controlled deep features selection framework for grape leaf diseases recognition. Expert Systems (2020)

20. Jiang, P., Chen, Y., Liu, B., He, D., Liang, C.: Real-time detection of apple leaf diseases using deep learning approach based on improved convolutional neural networks. IEEE Access 7, 59069–59080 (2019)

21. Ferentinos, K.P.: Deep learning models for plant disease detection and recognition of citrus deformities. Microsc. Res. Tech. 82, 1542–1556 (2019)

22. Sun, Y., et al.: Deep learning for plant identification in natural environment. Comput. Intell. and Neuroscience 2017, 1–6 (2017)

23. Khan, M.A., et al.: StomachNet: Optimal deep learning features fusion and correlation based feature selection. IEEE Access 7, 46261–46277, (2019)

24. Rashid, M., et al.: A sustainable deep learning framework for object recognition using multi-layers deep features fusion and selection. Sustainability 12, 5037 (2020)

25. Mälakä, U., et al.: Real time path finding for assisted living using deep learning. Journal of Universal Computer Science 24(4), 475–487 (2018)

26. Nasir, M.M., et al.: Deep learning-based classification of fruit diseases: An application for precision agriculture. CMC-Computers Materials & Continua 66, 1949–1962 (2021)

27. Muhammad, N., et al.: Severity recognition of aloe vera diseases using AI in sensor flow domain. CMC-Computers Materials & Continua 26, 2219–2226 (2021)

28. Safdar, A., et al.: Intelligent microscopic approach for identification and recognition of citrus deformities. Microsc. Res. Tech. 82, 1542–1556 (2019)

29. Dubey, S.R., Jalal, A.S.: Apple disease classification using color, texture and shape features from images. Signal, Image and Video Processing 10, 819–826 (2016)

30. Omrani, E., et al.: Potential of radial basis function-based support vector regression for apple disease detection. Measurement 55, 512–519 (2014)
31. Shuaibu, M., et al.: Detection of apple marssonina blotch disease using particle swarm optimization. Trans. ASABE 60, 303–312 (2017)
32. Barbedo, J.G.A.: A new automatic method for disease symptom segmentation in digital photographs of plant leaves. European Journal of Plant Pathol. 147, 349–364 (2017)
33. Khan, M.A., et al.: Skin lesion segmentation and classification: A unified framework of deep neural network features fusion and selection. Expert Systems e12497 (2019)
34. Abdulla, W.: Mask r-cnn for object detection and instance segmentation on keras and tensorflow (2017)
35. Lin, T.-Y., et al.: Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2117–2125. IEEE, Piscataway, NJ (2017)
36. Uijlings, J.R., et al.: Selective search for object recognition. Int. J. Comput. Vision 104, 154–171 (2013)
37. Neubeck, A., Van Gool, L.: Efficient non-maximum suppression. In: 18th International Conference on Pattern Recognition, pp. 850–855. IEEE, Piscataway, NJ (2006)
38. Jaderberg, M., Simonyan, K., Zisserman, A.: Spatial transformer networks. In: Advances in Neural Information Processing Systems, pp. 2017–2025. MIT Press, Boston, MA (2015)
39. Akiba, T., Suzuki, S., Fukuda, K.: arXiv:1711.04325, (2017)
40. Deng, J., et al.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255. IEEE, Piscataway, NJ (2009)
41. Mohanty, S.P., Hughes, D.P., Salathé, M.: Using deep learning for image-based plant disease detection. Front. Plant Sci. 7, 1419 (2016)
42. Jhuria, M., Kumar, A., Borse, R.: Image processing for smart farming: Detection of disease and fruit grading. In: 2013 IEEE Second International Conference on Image Information Processing, pp. 521–526. IEEE, Piscataway, NJ (2013)
43. Kawasaki, Y., Uga, H., Kagawa, S., Iyatomi, H.: Basic study of automated diagnosis of viral plant diseases using convolutional neural networks. In: International Symposium on Visual Computing, pp. 638–645. Springer, Berlin (2015)
44. Liu, B., Zhang, Y., He, D., Li, Y.: Identification of apple leaf diseases based on deep convolutional neural networks. Symmetry 10, 11 (2018)
45. Fujita, E., et al.: Basic investigation on a robust and practical plant diagnostic system. In: 2016 15th IEEE International Conference on Machine Learning and Applications, pp. 989–992. IEEE, Piscataway, NJ (2016)
46. Wang, G., Sun, Y., Wang, J.: Automatic image-based plant disease severity estimation using deep learning. Comput. Intell. Neurosci. 2017, 1–8 (2017)