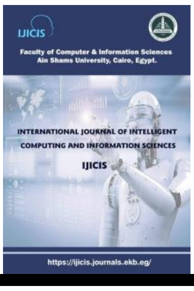
IJICIS, Vol.24, No.4, 60-72
DOI: 10.21608/ijicis.2024.333285.1363

International Journal of Intelligent Computing and Information Sciences

https://ijicis.journals.ekb.eg/


IJICIS
Faculty of Computer & Information Sciences Ain Shams University, Cairo, Egypt.
INTERNATIONAL JOURNAL OF INTELLIGENT COMPUTING AND INFORMATION SCIENCES
IJICIS
https://ijicis.journals.ekb.eg/

HYBRID CNN-RNN ARCHITECTURE FOR ACCURATE TOMATO DISEASE WITH XCEPTION-GRU

Batool Anwar*

Department of Computer Science, Faculty of Computer and Information Science, Ain Shams University,
Abbassia, Postal Code: 11566, Cairo, Egypt
ORCID: https://orcid.org/0009-0000-0798-7803
Batool.anwar@cis.asu.edu.eg

Mohamed M. Morsey

Department of Computer Science, Faculty of Computer and Information Science, Ain Shams University,
Abbassia, Postal Code: 11566, Cairo, Egypt
ORCID: https://orcid.org/0009-0005-6181-9057
mohamed.mabrouk@cis.asu.edu.eg

Islam Hegazy

Department of Computer Science, Faculty of Computer and Information Science, Ain Shams University,
Abbassia, Postal Code: 11566, Cairo, Egypt
ORCID: https://orcid.org/0000-0002-1572-463X
islheg@cis.asu.edu.eg

Zaki T. Fayed

Department of Computer Science, Faculty of Computer and Information Science, Ain Shams University,
Abbassia, Postal Code: 11566, Cairo, Egypt
ORCID: https://orcid.org/0009-0003-4090-6026
zaki.taha@cis.asu.edu.eg

Taha El-Arif

Department of Computer Science, Faculty of Computer and Information Science, Ain Shams University,
Abbassia, Postal Code: 11566, Cairo, Egypt
ORCID: https://orcid.org/0009-0006-0896-6063
tahaelarif@cis.asu.edu.eg

Received 2024-11-02; Revised 2024-11-02; Accepted 2024-12-16

Abstract: *Plant diseases and pernicious insects are a considerable threat in the agriculture sector. Therefore, early detection and diagnosis of these diseases are essential. The ongoing development of profound deep learning methods has greatly helped in the detection of plant diseases, granting a vigorous tool with exceptionally precise outcomes but the accuracy of deep learning models depends on the volume and the quality of labeled data for training. In this paper, we have proposed a deep learning-based method for tomato disease detection that utilizes the ensemble convolutional neural network (CNN) and recurrent neural network (RNN) architecture. The proposed model is known as Xception-GRU as it begins with the Xception pre-trained model and followed by the GRU layers. Thereafter, the Xception-GRU model is trained on synthetic and real images using transfer learning to classify the tomato leaves images into ten categories of diseases. Three different classifiers are used on the features extracted from the Xception-GRU model. These classifiers are the k-nearest neighbor (KNN), multi-layer perceptron (MLP), and support vector machine (SVM). The proposed model has been trained and tested extensively on publicly available PlantVillage dataset. The proposed method achieved an accuracy of 100%, 98.79%, 99.85%, and 100% for tomato leaf image diseases classification into (Early, Late Blight, and Healthy), (Late and Early Blight), (Late Blight and Healthy),*

***Corresponding Author**: Batool Anwar

Computer Science Department, Faculty of Computer and Information Science, Ain Shams University, Cairo, Egypt

Email address: Batool.anwar@cis.asu.edu.eg

and (Early and Healthy). The proposed approach shows its superiority over the existing methodologies..

Keywords: *Plant Diseases, Pernicious insects, Xception-GRU model, Synthetic images, Ensemble architecture*

1. Introduction

Agriculture has long been the main source of income for many in several developing countries, but the widespread commercialization of farming has had significant environmental impacts. One major concern is the detection of plant diseases, as early identification can prevent their spread and mitigate economic losses. Plant diseases can range from mild manifestations to the complete destruction of crops, posing a serious threat to the agricultural economy [1].
Deep learning, a powerful technique that combines data analysis with image processing, has emerged as a highly accurate tool in various fields. Its applications include object detection, features recognition, and biomedical image classification and segmentation. In agriculture, deep learning, particularly Convolutional Neural Networks (CNNs), is increasingly used for plant disease detection and classification. CNN architectures like AlexNet and GoogLeNet have shown promising results in this regard [2].
However, the performance of CNN models heavily relies on the quality and quantity of available training data. Often, existing datasets for tomato plant diseases lack diversity and size, leading to challenges such as overfitting and poor generalization [3]. To address this, researchers employ data augmentation techniques like affine and perspective transformations. When these techniques are insufficient, Generative Adversarial Networks (GANs) are utilized to generate synthetic images [4].
In this study, we propose a deep learning framework for tomato plant disease recognition based on analyzing tomato leaf images [5, 6, and 7]. This approach aims to assist farmers by enabling disease classification simply through images of affected leaves, avoiding the need for costly expert analysis. Subsequently, a Xception-GRU model is trained on both real and synthetic tomato plant images for disease detection [8, 9, and 10].
The paper is structured as follows: Section 2 reviews existing methods, Section 3 outlines the methodology including Xception-GRU model, Section 4 summarizes the dataset, experimental setup, and results, including performance comparisons with existing methods and pre-trained CNN models. Conclusions are presented in Section 5.

2. Related Work

In agriculture, the presence of plant diseases poses a significant threat, leading to economic losses. Researchers have proposed various techniques to combat these infections, with recent attention focusing on deep learning methods, particularly convolutional neural networks (CNNs), for disease detection and classification. All of the following studies are performed on a plant village dataset. The PlantVillage dataset consists of 54303 healthy and unhealthy leaf images divided into 38 categories by species and disease .
Akila and Deepan (2018) [11] introduced a deep learning model utilizing Faster Region-based Convolutional Neural Network (R-CNN) [12], Region-based Fully Convolutional Network (R-FCN), and Single Shot Detector (SSD) [13, 14] for detecting plant diseases. Their dataset comprised images of

fruit, vegetable, cereal, and cash crops, augmented with techniques like image rotations and brightness adjustments achieving an accuracy of 94%.
Adhikari et al. (2018) [15] developed a model specifically for detecting tomato plant diseases through image processing. Their dataset included images of Gray spot, Late Blight, and Bacterial Canker, sourced from both the internet and farms [16]. Their CNN model achieved an accuracy of 89% on the PlantVillage dataset and 76% on their own dataset.
Fuentes et al. (2017) [17] proposed a system capable of identifying nine types of diseases and pests in tomato plants, including disease classification and localization. They employed detectors like R-FCN, Faster R-CNN, and SSD with feature extractors such as VGG-16 and ResNet50. The model achieved mean Average Precision (mAP) ranging from 75.37% to 85.98%.
Ashqar and Abu-Naser (2018) [18] developed a CNN model to identify five diseases in tomato plants, achieving high accuracies of 99.84% and 95.54% on full-color and grayscale images, respectively, using 9,000 images from the PlantVillage dataset.
Zhang et al. (2018) [19] proposed a CNN model for distinguishing eight types of tomato leaf diseases through transfer learning with pre-trained models like AlexNet, GoogLeNet, and ResNet, achieving a highest accuracy of 97.28% with ResNet using SGD.
Similarly, Durmus et al. (2017) [20] employed AlexNet and SqueezeNet to classify tomato plant images into ten classes, achieving accuracies of 95.65% and 94.3%, respectively, on the PlantVillage dataset.
Karthik et al. (2020) [21] utilized two deep learning architectures on the PlantVillage dataset to detect three diseases in tomato plants, achieving a highest accuracy of 98% with an attention-based residual CNN architecture.
Agarwal et al. (2020) [22] developed a CNN model to detect ten classes of tomato plant diseases with an overall accuracy of 91.2% on the PlantVillage dataset. Elhassouny and Smarandache (2019) [23] designed a smart mobile application using MobileNet to classify nine diseases of tomato plants, achieving an accuracy of 90.3% on 7,176 tomato leaf images from the PlantVillage dataset.[24]
Widiyanto et al. (2019) [25] developed a CNN model to identify four diseases in tomato plants along with healthy leaves, achieving an overall classification accuracy of 96.6% on 1,000 images per class from the PlantVillage dataset. Table 1 summarizes the various deep learning approaches used in the literature for detecting and classifying tomato plant diseases.

Table.1 Literature review on the studies presented for plant diseases detection

Authors	**Methodology**	**Accuracy**
Akila and Deepan (2018)	Faster R-CNN, R-FCN, SSD	94%
Adhikari et al. (2018)	CNN with YOLO architecture	89% (PlantVillage), 76% (own dataset)
Fuentes et al. (2017)	R-FCN, Faster R-CNN, SSD with VGG-16, ResNet50	75.37% - 85.98% (mAP)
Ashqar and Abu-Naser (2018)	CNN with ReLU activation, max-pooling, 4 conv layers	99.84% (full-color), 95.54% (grayscale)
Zhang et al. (2018)	CNN with transfer learning (AlexNet, GoogLeNet, ResNet)	97.28% (ResNet with SGD)
Durmus ¸ et al. (2017)	AlexNet, SqueezeNet	95.65% (AlexNet), 94.3% (SqueezeNet)
Karthik et al. (2020)	Attention-based residual CNN, feed-forward CNN	98% (Attention-based residual CNN)
Agarwal et al. (2020)	CNN with 3 conv, 3 max-pooling, 2 fully connected layers	91.2%
Elhassouny and Smarandache (2019)	MobileNet	90.3%

Widiyanto et al. (2019)	CNN	96.6%

3. Methodology

In this Section, we will examine the suggested technique. The schematic representation of the suggested approach is depicted in Figure 1. The proposed method can be segmented into two components: extracting the features using the Xception-GRU, while the second component is the classification based on k-nearest neighbor (KNN), multi-layer perceptron (MLP), support vector machine (SVM). Subsequently, a pre-existing Xception-GRU model is refined on tomato leaf images to classify diseases.

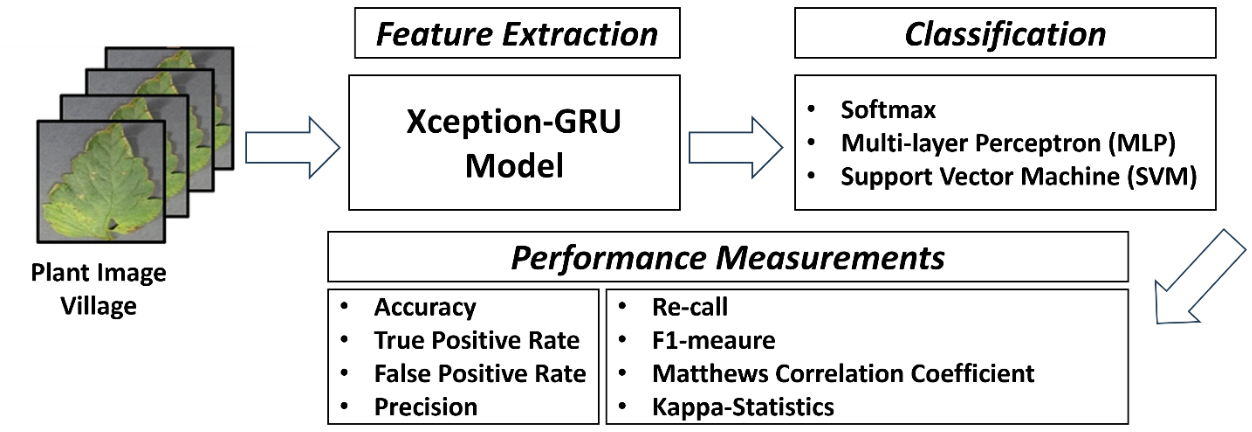


Figure.1 Proposed Overall Methodology

3.1. Data Acquisition

The PlantVillage dataset contains 54,305 single leaf images (256 px2 , RGB) from 14 crop species [26]. There are 38 classes named as species disease or species healthy. The leaves were removed from the plant, placed against a grey or black background, and photographed outdoors with a single digital camera on sunny or cloudy days. In this paper, we used the version of the dataset shared by Mohanty et al., which contains the original images, grayscale images and images without the backgrounds [27].

3.2. Feature Extraction

Feature extraction is a fundamental concept in the field of machine learning and pattern recognition. It involves the process of transforming raw data into a set of features that are more informative and representative of the underlying patterns or characteristics present in the data. These features serve as inputs to machine learning algorithms for tasks such as classification, clustering, and regression.

The Xception model, a robust pre-trained model in deep learning, relies on grouped convolutional layers, also known as depth-wise convolutional layers. These layers divide input channels into groups, convolving the input within each group vertically and horizontally. This separation decouples spatial correlations and cross-channel influences in the feature map. The Xception model comprises 14 blocks with linear residual connections, excluding the first and last blocks. These blocks contain a total of 36 convolutional layers, organized into three flows: entry, middle, and exit. In the entry flow, features are derived from 8 primary convolutional layers followed by batch normalization and “Swish” activation [28].

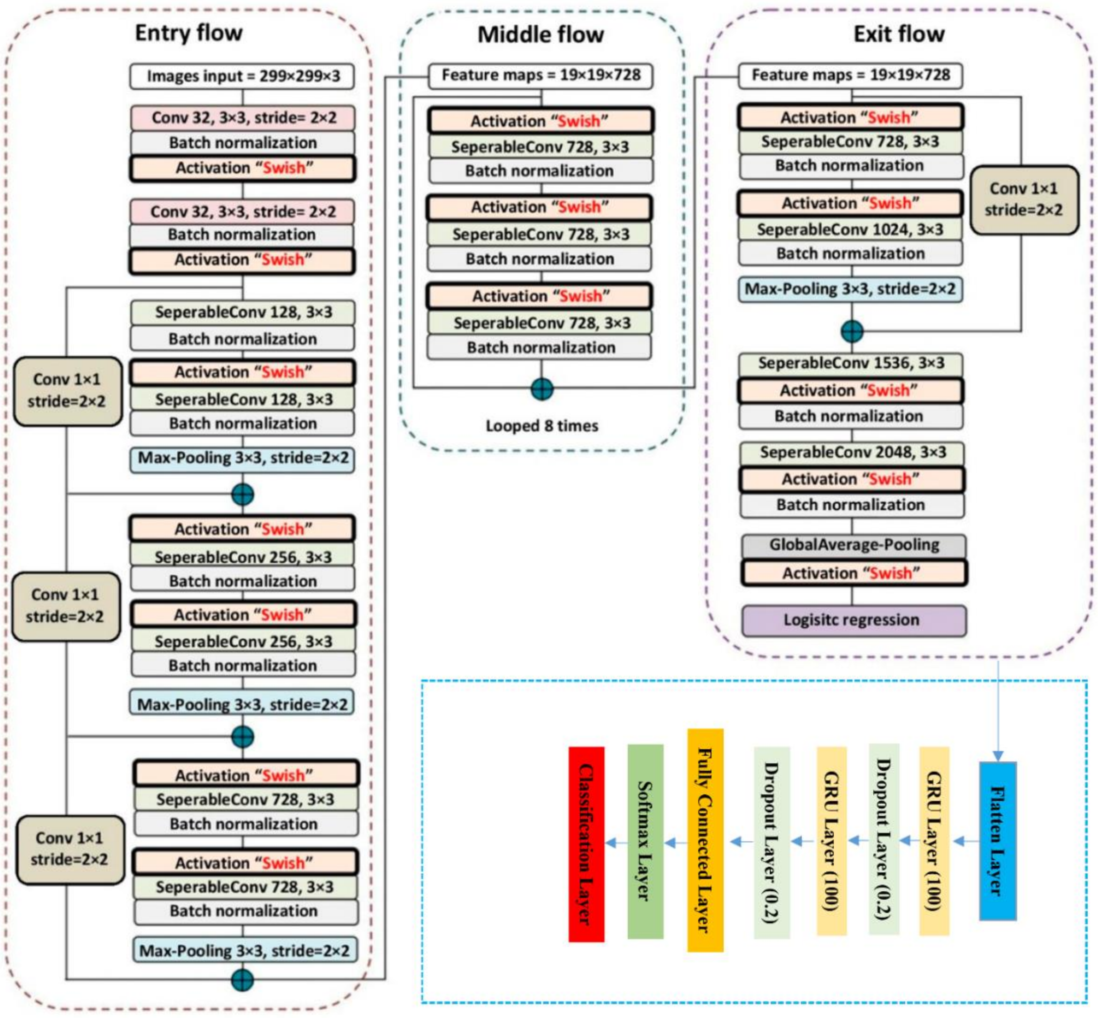


Figure.2 Xception-GRU Methodology for Feature Extraction

The first two layers are standard convolutional layers, while the remaining are separable (depth-wise) convolutional layers. Additionally, max-pooling layers are inserted after every two grouped convolutional layers in the entry flow, gradually increasing the number of filters from 32 to 728 to extract input features effectively. The middle flow, consisting of 24 depth-wise convolutional layers with 728 filters, extracts more complex features.

In the subsequent third flow, detailed features are extracted through 4 depth-wise convolutional layers, with filter numbers escalating to 786, 1024, 1536, and 2048, respectively. After the exit flow, the Xception model is followed by a global average pooling layer and a fully connected layer. The proposed Xception-GRU architecture transfers the fully connected layer's output from the Xception model to a stacked GRU model. This GRU model comprises a sequence input layer, 2 GRU layers, 2 dropout layers, a fully connected layer, softmax layer, and a classification layer [29].

Each GRU has 100 hidden units, and dropout layers have a dropout probability of 0.2. Figure 2 illustrates the Xception-GRU model, where input data traverse through the entry and middle flows, repeated eight times, before entering the exit flow. All standard and depth-wise convolutional layers are followed by batch normalization and ReLU layers, and depth-wise layers maintain a depth multiplier of 1.

3.3 Classification

This step is the final step of our methodology in which the result of the diagnosis will be determined with average accuracy. Three main classifiers are determined to examine the performance of the methodology, and these classifiers are Softmax, Multi-layer perceptron (MLP) and Support vector machine (SVM).

3.3.1 K-nearest neighbor (KNN) Classifier

The K-Nearest Neighbors (KNN) algorithm is a non-parametric, instance-based learning method used for classification and regression tasks. It works based on the assumption that similar data points tend to belong to the same class or have similar output values. In the context of classification, KNN assigns a class label to a new data point based on the majority class among its k nearest neighbors. One of the key components of the KNN algorithm is the distance metric used to measure the similarity between data points. The most commonly used distance metric in KNN is the Euclidean distance, although other distance metrics such as Manhattan, Minkowski, or Cosine similarity [30].
In the KNN algorithm with Euclidean distance, when a new data point is to be classified, the distances between the new data point and all other data points in the training set are computed. The k nearest neighbors of the new data point are then identified based on the smallest Euclidean distances. Finally, the majority class among the k nearest neighbors is assigned as the predicted class label for the new data point.
KNN with Euclidean distance is intuitive and easy to implement, making it a popular choice for classification tasks, especially in cases where the decision boundary is irregular or nonlinear. However, it has some limitations, such as computational inefficiency with large datasets and sensitivity to the choice of the number of neighbors (k) and the distance metric. Additionally, it assumes that all features contribute equally to the distance calculation, which may not always be appropriate for all datasets. Despite these limitations, KNN with Euclidean distance remains a valuable tool in the machine learning toolkit, particularly for its simplicity and effectiveness in certain scenarios.

3.3.2 Multi-layer perceptron (MLP) Classifier

In this model, the KNN classifier is replaced by the MLP. MLP is a feedforward ANN that maps the inputs onto a set of output categories. ANNs consist of neurons that operate in parallel, i.e., parallel processing, which is an important advantage. The classification function of the neural network is dependent on the product of the weights between and the biases within the neurons. The training is based on backpropagation. The network weights are adapted and updated until the operation produces the desired output [31]. Mathematically, the interval activities of the neurons in the network are given by:

$$Y_k = f\left(\sum_{j=1}^{n_k} w_{kj}\left(f \sum_{i=1}^{d} w_{ji}x_i + w_{j0}\right) + w_{k0}\right) \quad (1)$$

where x_I represent features fed to the input layer; Y_k, the final result of the neuron; and j, the index of the neuron. The weight between the input (i) and the hidden layer neuron (j) is W_{ji}, while W_{kj} is the weight between the j and the output (k). w_{j0}, w_{k0}, and n_k are the biases of the hidden, output, and the last neurons, respectively. f (.) is the activation function for the hidden and output layers, which are log-sigmoid for the hidden and linear activation layers of the output layer. The full network has 28 layers consisting of 25 CNN layers followed by three layers of the ANN, which compose the MLP. The CNN performs feature extraction with momentum backpropagation. The features are input to the MLP for classification. There is an input layer, where the number of neurons corresponds to the number of features output by the CNN, i.e., 1,000. The hidden layer contains 2,000 neurons, which is the product of the number of classes (4) and the number of features (1,000) divided by 2. The output layer consists of four neurons, which yield the final classification results for the four classes. There are logsig activation functions in the hidden units, while the output unit contains a linear function. Cross-entropy loss function was used to measure the error between the actual and the desired output.

3.3.3 Support Vector Machine (SVM) Classifier

In this hybrid model, CNN is combined with the efficient shallow SVM classifier, i.e., SVM replaces the classification layer of the CNN. The SVM classifier was developed for binary classification [32] and aims to find the optimal hyperplane $\mathrm{f(w,x) = w.x + b}$ that separates two classes in a given dataset. SVM learns the parameters w and b by solving an optimization problem, as given by:

$$Minimize: \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{p} E_i$$
$$s.t \begin{cases} y_i(w \cdot x + b) \geq 1 - E_i \\ E_i \geq 0, i = 1 \ldots\ldots. p(No. of Samples) \end{cases} \tag{2}$$

where $\|w\|^2$ represent the Euclidean norm; C, the penalty parameter that governs maximization of margin and minimization of classifications error; E, the cost function; and y, the actual label. The best hyperplane is determined by minimizing the cost function while taking into account the two conditions of maximizing the margin and minimizing the error. This optimization problem can be expressed as a quadratic programming problem subject to constraints, and the solution for the discriminant function can be expressed as:

$$f(x) = \sum_i \alpha_i y_i k(x_i.x) + b \tag{3}$$

where $k(x_i.x)$ represents the kernel function that maps the input data to a higher dimensional space for maximization of the margin; and α_i the Lagrange Multipliers. In our case, $k(x_i.x)$ is the polynomial of degree d, i.e., $k(x_i.x) = (1 + x_i \cdot x)^d$. To build a multiclass classifier based on SVM, error-correcting output code (ECOC) was applied, which reduced the multiclass classification problem to a set of binary problems using the one-against-one (OAO) or one-against-all (OAA) strategies. In our case, we used OAO, which would involve four classes with SVM based on six datasets. Finally, the four classifiers were combined using the ECOC pattern, the strategy of the combination being based on a one-versus-one coding design.

4. Experimental Setup and Discussion

The model was implemented using MATLAB, which supplied all the requisite libraries, on a laptop that has these aspects which are Intel Core i7-8565U 8 MB cache, 2.81 GHz central processing unit, followed with a 12 Gram and a graphics card NVIDA GeForce MX 310 M 4 GB.

4.1 Training Parameters Setting

The table 2 presents the results of experiments conducted using different classifiers (KNN, MLP, and SVM) for classifying tomato leaf images into three categories: Early (E), Healthy (H), and Late (L). Each row in the table represents a specific experiment comparing two classes or all three classes.

Here's a breakdown of the metrics and what they signify:

- A: Represents the classes being compared in each experiment (e.g., E vs H vs L, E vs H, E vs L, H vs L).
- TP: True Positives, the number of correctly classified instances for the specified class or classes.
- FP: False Positives, the number of incorrectly classified instances as positive.
- K: The number of classes involved in the experiment.
- TPR: True Positive Rate (Sensitivity), the proportion of actual positive instances correctly identified by the model.
- FPR: False Positive Rate, the proportion of actual negative instances incorrectly classified as positive.
- P: Precision, the proportion of correctly predicted positive instances out of all instances predicted as positive.
- R: Recall (Sensitivity), the proportion of actual positive instances correctly identified by the model.
- F1: F1 Score, the harmonic mean of precision and recall, providing a balanced measure between the two.
- MCC: Matthews Correlation Coefficient, a measure of the quality of binary classifications, considering true and false positives and negatives.
- ROC: Receiver Operating Characteristic, a graphical plot that illustrates the diagnostic ability of a binary classifier system.
- PRC: Precision-Recall Curve, a graph that shows the trade-off between precision and recall for different thresholds.

Here are some observations based on the provided results:

All classifiers achieved perfect accuracy (100%) when classifying among all three classes (E vs H vs L) as well as when comparing only Early and Healthy classes (E vs H). This suggests that the models performed exceptionally well in distinguishing between these classes. When comparing Early vs Late (E vs L) classes, the accuracy slightly decreased to around 98.79% for KNN and MLP, and 98.45% for SVM. This indicates that distinguishing between these two classes might be slightly more challenging than distinguishing between Early and Healthy classes.

The classifiers performed similarly well in distinguishing between Healthy and Late (H vs L) classes, achieving accuracy rates of around 99.71% to 99.85%. This suggests that the models were highly effective in differentiating between Healthy and Late stages of tomato plant diseases. In terms of other metrics such as precision, recall, F1 score, and Matthews Correlation Coefficient, the classifiers generally performed very well across all experiments, indicating robust performance in classifying tomato leaf images into different disease stages. Overall, the results suggest that the classifiers, particularly SVM, MLP, and KNN, are highly effective in classifying tomato leaf images into different disease stages, with high accuracy and performance across various metrics.

Table.2 Performance Classifers' results on the performed experiments using Xception-GRU model

Classifiers	**Experiments**												
		A	**TP**	**FP**	**K**	**TPR**	**FPR**	**P**	**R**	**F1**	**MCC**	**ROC**	**PRC**
Softmax	E vs H vs L	100	901	0	1.000	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000
	E vs H	100	519	0	1.000	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000
	E vs L	98.79	575	7	0.973	0.988	0.018	0.988	0.988	0.988	0.973	0.985	0.981
	H vs L	99.71	698	2	0.994	0.997	0.002	0.997	0.997	0.997	0.994	0.997	0.996
MLP	E vs H vs L	100	901	0	1.000	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000
	E vs H	100	519	0	1.000	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000
	E vs L	98.62	574	8	0.969	0.986	0.019	0.986	0.986	0.986	0.969	1.000	1.000
	H vs L	99.85	699	1	0.997	0.999	0.001	0.999	0.999	0.999	0.997	0.999	0.998
SVM	E vs H vs L	99.8	900	1	0.999	0.001	0.999	0.999	0.999	0.999	0.998	0.998	0.999
	E vs H	100	519	0	1.000	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000
	E vs L	98.45	574	8	0.969	0.986	0.010	0.987	0.986	0.986	0.970	0.988	0.981
	H vs L	99.85	699	1	0.997	0.999	0.001	0.999	0.999	0.999	0.997	0.999	0.998

4.2 Confusion Matrices

A confusion matrix is a table used to evaluate the performance of a classification model by summarizing the counts of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions made by the model on a dataset with known true labels. It provides a detailed breakdown of how well the model is performing across different classes.
Here's a brief description of each element in a confusion matrix:
True Positive (TP): The number of instances correctly predicted as positive by the model.
True Negative (TN): The number of instances correctly predicted as negative by the model.
False Positive (FP): The number of instances incorrectly predicted as positive by the model (also known as Type I error).
False Negative (FN): The number of instances incorrectly predicted as negative by the model (also known as Type II error).
A confusion matrix is typically organized into a square matrix, where the rows represent the actual classes or labels, and the columns represent the predicted classes by the model. Each cell in the matrix represents the count of instances for a specific combination of actual and predicted classes. Figures. 3 to 7 show the confusion matrices of the four experiments performed using Xception-GRU model based on KNN, MLP, and SVM classifiers.

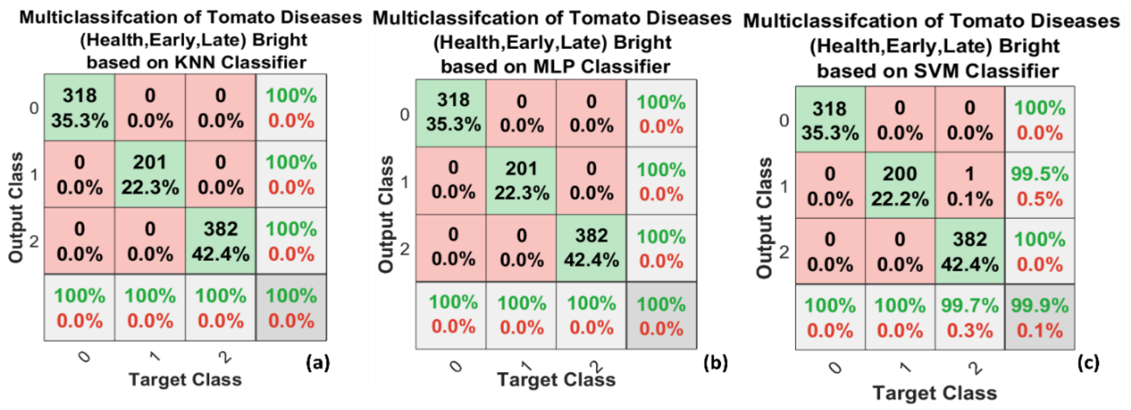


Figure.3 Confusion Matrices of Multi-Classification experiment (Health vs Early vs Late) Tomato Diagnosis

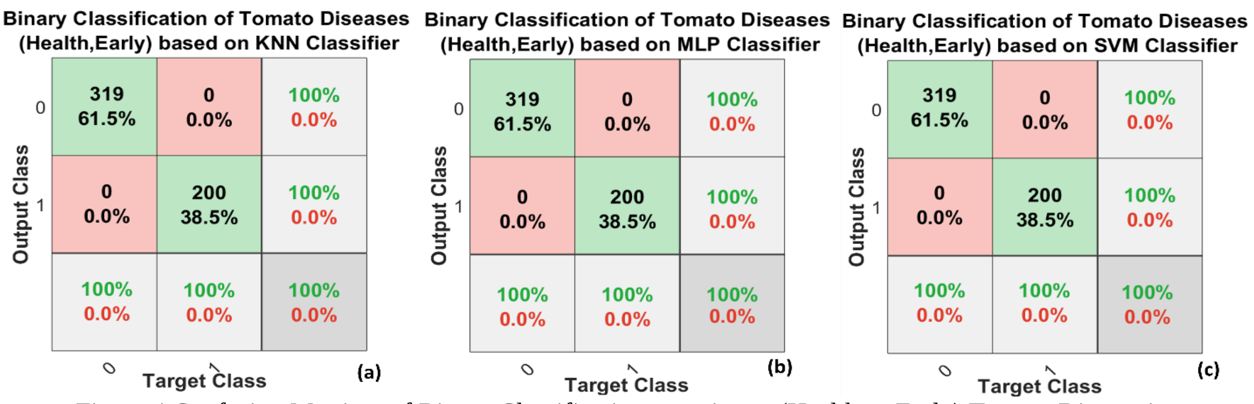


Figure.4 Confusion Matrices of Binary Classification experiment (Health vs Early) Tomato Diagnosis

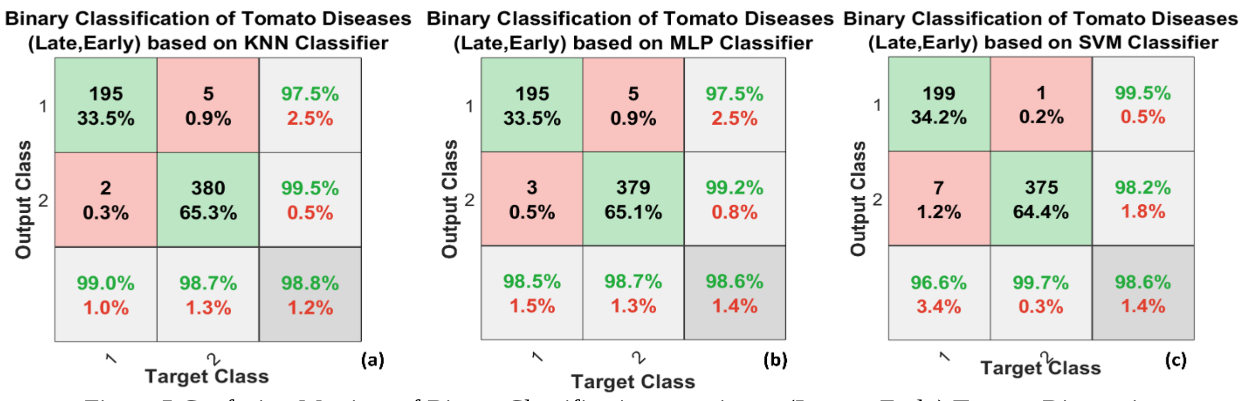


Figure.5 Confusion Matrices of Binary Classification experiment (Late vs Early) Tomato Diagnosis

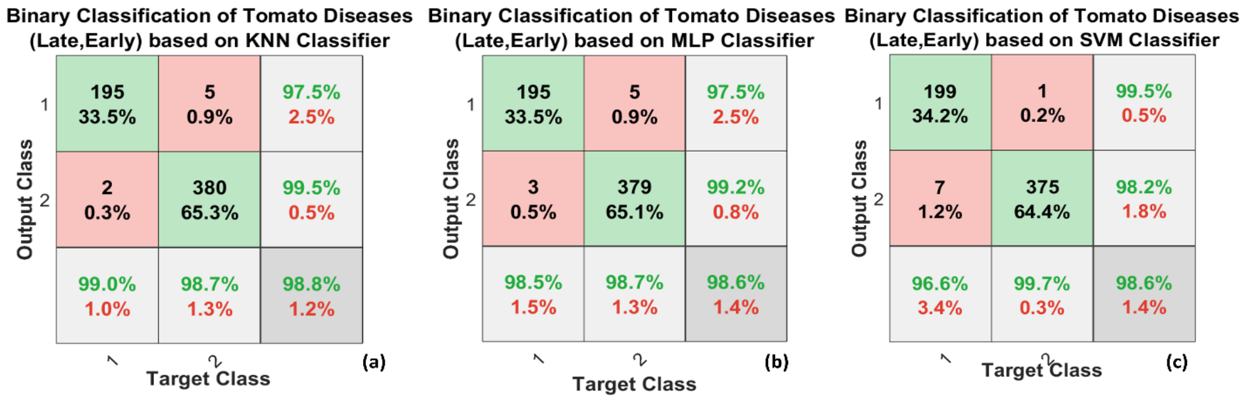


Figure.6 Confusion Matrices of Binary Classification experiment (Late vs Early) Tomato Diagnosis

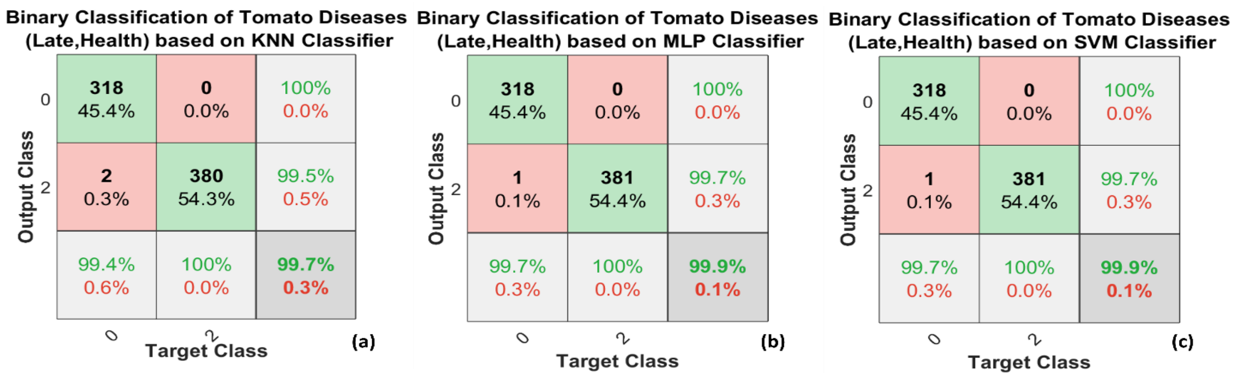


Figure.7 Confusion Matrices of Binary Classification experiment (Late vs Health) Tomato Diagnosis

5. Conclusion

Deep learning techniques have demonstrated significant potential in the realm of plant disease detection. Our study introduces a novel deep learning-based approach for identifying tomato plant diseases from images of tomato leaves. The proposed methodology is based on the ensemble of the Xception and the gated recurrent unit layers. This ensemble leads to the generation of robust features that can be classified easily using various classifiers. Our proposed model achieves impressive accuracies of 100%, 98.79%, and 99.85% on the original PlantVillage dataset for 3-class and 2-class classification tasks, respectively. Moreover, when utilizing the augmented dataset comprising original PlantVillage images. Furthermore, our experimental results demonstrate the superiority of our proposed method over existing approaches.

References

1. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z., 2016. Rethinking the inception architecture for computer vision, in. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2818–2826.
2. Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016. You only look once: Unified, real-time object detection, in. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788.

3. Abdel-Hamid, O., Mohamed, A.-R., Jiang, H., Deng, L., Penn, G., Yu, D., 2014. Convolutional neural networks for speech recognition. IEEE/ACM Trans. Audio, Speech, Language Process. 22 (10), 1533–1545.
4. Gour, M., Jain, S., Agrawal, R., 2019. Deeprnnetseg: Deep residual neural network for nuclei segmentation on breast cancer histopathological images. In: International Conference on Computer Vision and Image Processing. Springer, pp. 243–253.
5. Gour, M., Jain, S., Sunil Kumar, T., 2020. Residual learning based cnn for breast cancer histopathological image classification. Int. J. Imaging Syst. Technol.
6. Gour, M., Jain, S., 2020. Stacked convolutional neural network for diagnosis of covid-19 disease from x-ray images, arXiv preprint arXiv:2006.13817.
7. Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (Eds.), Advances in Neural Information Processing Systems 25. Curran Associates, Inc., pp. 1097–1105
8. Adhikari, S., Saban Kumar, K., Balkumari, L., Shrestha, B., Baiju, B., 2018. Tomato plant diseases detection system using image processing. In: 1st KEC Conference on Engineering and Technology, Lalitpur, vol. 1, pp. 81–86.
9. Mirza, M., Osindero, S., 2014. Conditional generative adversarial nets, arXiv preprint arXiv:1411.1784.
10. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q., 2017. Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4700–4708.
11. Akila, M., Deepan, P., 2018. Detection and classification of plant leaf diseases by using deep learning algorithm. Int. J. Eng. Res. Technol 6, 2–7
12. Ren, S., He, K., Girshick, R., Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems, pp. 91–99.
13. Dai, J., Li, Y., He, K., Sun, J., 2016. R-fcn: Object detection via region-based fully convolutional networks. In: Advances in Neural Information Processing Systems, pp. 379–387.
14. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., Berg, A.C., 2016. Ssd: Single shot multibox detector. In: European Conference on Computer Vision. Springer, pp. 21–37.
15. Agarwal, M., Singh, A., Arjaria, S., Sinha, A., Gupta, S., 2020. Toled: Tomato leaf disease detection using convolution neural network. Procedia Comput. Sci. 167, 293–301.
16. Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016. You only look once: Unified, real-time object detection, in. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788.
17. Fuentes, A., Yoon, S., Kim, S.C., Park, D.S., 2017. A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition. Sensors 17 (9), 2022.
18. Ashqar, B.A., Abu-Naser, S.S., 2018. Image-based tomato leaves diseases detection using deep learning.
19. Zhang, K., Wu, Q., Liu, A., Meng, X., 2018. Can deep learning identify tomato leaf disease? Adv. Multimedia.
20. Durmus ¸, H., Günes ¸, E.O., Kırcı, M., 2017. Disease detection on the leaves of the tomato plants by using deep learning. In: 2017 6th International Conference on Agro- Geoinformatics. IEEE, pp. 1–5.
21. Karthik, R., Hariharan, M., Anand, S., Mathikshara, P., Johnson, A., Menaka, R., 2020. Attention embedded residual cnn for disease detection in tomato leaves. Appl. Soft Comput. 86, 105933.
22. Agarwal, M., Singh, A., Arjaria, S., Sinha, A., Gupta, S., 2020. Toled: Tomato leaf disease detection using convolution neural network. Procedia Comput. Sci. 167, 293–301.

23. Elhassouny, A., Smarandache, F., 2019. Smart mobile application to recognize tomato leaf diseases using convolutional neural networks. In: 2019 International Conference of Computer Science and Renewable Energies (ICCSRE), IEEE, pp. 1–4.
24. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H., 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications, arXiv preprint arXiv:1704.04861.
25. Widiyanto, S., Fitrianto, R., Wardani, D.T., 2019. Implementation of convolutional neural network method for classification of diseases in tomato leaves. In: 2019 Fourth International Conference on Informatics and Computing (ICIC). IEEE, pp. 1–5.
26. Plant Village Dataset: https://www.kaggle.com/datasets/emmarex/plantdisease Last accessed 28-10-2024.
27. Salim, F., Saeed, F., Basurra, S., Qasem, S. N., & Al-Hadhrami, T. (2023). DenseNet-201 and Xception pre-trained deep learning models for fruit recognition. *Electronics*, *12*(14), 3132.
28. Dey, R., & Salem, F. M. (2017, August). Gate-variants of gated recurrent unit (GRU) neural networks. In *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)* (pp. 1597-1600). IEEE.
29. Liao, Y., & Vemuri, V. R. (2002). Use of k-nearest neighbor classifier for intrusion detection. *Computers & security*, *21*(5), 439-448.
30. Windeatt, T. (2006). Accuracy/diversity and ensemble MLP classifier design. *IEEE Transactions on Neural Networks*, *17*(5), 1194-1211.
31. Keerthi, S. S., Shevade, S. K., Bhattacharyya, C., & Murthy, K. R. K. (2001). Improvements to Platt's SMO algorithm for SVM classifier design. *Neural computation*, *13*(3), 637-649.