Mehran University Research Journal of Engineering and Technology

https://doi.org/10.22581/muet1982.2303.06

2023, 42(3) 52-58

Deep learning image-based automated application on classification of tomato leaf disease by pre-trained deep convolutional neural networks

ReddyPriya Madupuri [a], Dinesh Reddy Vemula [a], Anil Carie Chettupally [a], Abdur Rashid Sangi [b,*], Pallam ravi [c]

[a] *Department of Computer Science and Engineering, SRM University-AP,Amaravati, Guntur India*

[b] *Department of Computer Science, College of Science and Technology, Wenzhou-Kean University, Ouhai, Wenzhou Zhejiang China*

[c] *Department of Computer Science and Engineering, Anurag University, Hyderabad India*

[*] Corresponding author: Abdur Rashid Sangi, Email: sangi_bahrian@yahoo.com

Received: 31 January 2023, Accepted: 26 June 2023, Published: 01 July 2023

K E Y W O R D S

Convolutional Neural Networks

Machine Learning

Image Classification

A B S T R A C T

The agriculture sector is one of the major sectors in India. India is well known for the production of various varieties of spices, fruits, vegetables, herbs, etc. Along with the pollution, the diseases that are affecting plants are increasing and there are various reasons for this. Tomato is one of the high-demand crops in the market and is produced in large quantities. There are many diseases that tomatoes get affected by because of the virus, fungus, bacteria, etc. In this project, we proposed a model to identify the diseases of tomato plants using images of tomato plant leaves. Our main goal is to develop a good model with decent accuracy and a mobile application that works with or without the internet for users, especially farmers. The Convolution Neural Network-based approach is used to create the model for this project. This proposed system model gives 98 % accuracy and that model is converted to the TF Lite model which is used in the application. This application can precisely predict the disease of the tomato leaf and suggest the treatment for it.

1. Introduction

Access India is a country known for agriculture. With 38% of the land area is suitable for agriculture. And all over the world, India is the second-largest producer of both rice and wheat. Cereals have the highest market for almost 46% of the Indian Agriculture market. Solanum Lycopersicum which is also known as Tomato made India rank second on the list of nations producing it with 852 thousand hectares of cultivation area[1-3]. Tomato is in second place for the crops getting affected with 12.8% of incidents. Generally, tomato grows on almost any well-drained soil, and it is a Rabi crop in plain regions whereas in hilly regions it can be grown in summer and rainy season [4]. We know that Andhra Pradesh has the highest tomato crop production with 21 million metric tons, and it is the second most state which is getting affected with 13.9% of incidents [5,6]. Along with production, the frequency of crops getting affected by diseases is also increasing. There are different ways to get diseases and all of them can't be identified with the farmer's expertise. In the early days, farmers used to monitor the plants manually and treat them according to that observation [7]. This requires a lot of work and time, but sometimes the

prediction of diseases can't be accurate. Deep learning is widely used in all sectors and with the help of this, we can predict diseases using visually observable patterns[8,9]. Monitoring the crops is very important and deep learning made it easy to work with. Using the images of various diseases and deep learning we can predict the disease with good accuracy[10,11]. Adding a mobile application to it will become handy for farmers to work. Mobile is developed to click or choose images from mobile and predict the disease. Along with those symptoms and treatments are provided for each disease to provide the basic knowledge to newbies in agriculture.

Objectives

The objectives of the proposed project are as below:

- To design an efficient system that may detect tomato diseases based on leaf pictures and can be used in farms and nurseries.
- An inference is performed using the TensorFlow Lite Java API.
- To develop an app that can work in places without internet connection so which will be helpful in farms and low network bandwidth places.
- To provide users with disease symptoms and their treatments.

2. Background and Literature Survey

There has been considerable research on using deep learning for automated disease detection and classification in various crops, including tomato plants. In this literature survey, we will focus on recent studies that use pre-trained deep convolutional neural networks (CNNs) for tomato leaf disease classification. [3] propose a Tomato Disease Classification Using Convolutional Neural Networks. This study used a pre-trained CNN called VGG-16 to classify tomato leaf diseases from images. The authors achieved an accuracy of 97.78% on a dataset of 10,000 tomato leaf images. [4] propose a Deep Learning for Automated Detection and Classification of Tomato Disease. In this study, the authors used a pre-trained CNN called AlexNet to classify four types of tomato diseases. The dataset contained 5,000 images and achieved an accuracy of 98.3%. [5] propose a Deep Learning Based Tomato Disease Recognition Using MobileNet and Inception-v3 Architectures. This study compared two pre-trained CNNs, MobileNet and Inception-v3, for tomato disease classification. The authors achieved an accuracy of 98.67% on a dataset of 15,000 tomato leaf images. [6] proposed a Tomato Leaf Disease Recognition Using Pre-trained Convolutional Neural Networks. The authors used pre-trained CNNs, including VGG-16, Inception-v3, and ResNet-50, to classify tomato leaf diseases. They achieved an accuracy of 97.45% on a dataset of 4,998 images. [7] proposed a Automated Detection and Classification of Tomato Leaf Diseases Using Deep Convolutional Neural Networks. This study used a pre-trained CNN called GoogLeNet to classify tomato leaf diseases. The dataset contained 10,250 images, and the authors achieved an accuracy of 96.7%. Overall, these studies demonstrate the effectiveness of pre-trained deep CNNs for tomato leaf disease classification. The accuracy achieved by different models and datasets varied, but they all showed promising results. Further research is needed to develop more robust and accurate models that can handle various environmental conditions for tomato leaf prediction. In this paper, we use a dataset from different soil types along with different climatic conditions to predict the tomato leaf disease.

3. Methodology

The system architecture of this work is illustrated by the below block diagram.

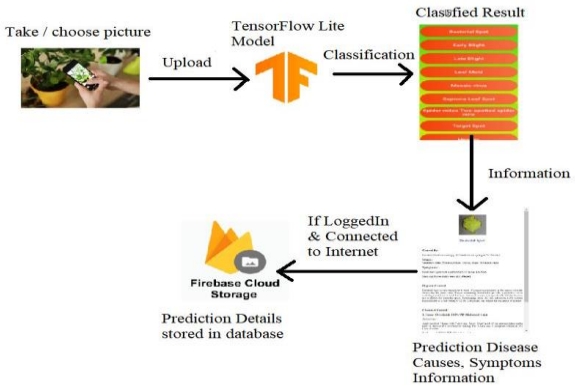


Fig. 1. Block diagram of the proposed model

3.1 Dataset

The tomato leaf disease images are taken from the plant leaf village dataset. This dataset consists of 14529 images belonging to 10 different classes. All these images belong to RGB Color space. Sample images from each class along with the disease names are depicted as follows (refer to Fig. 2). Here in our project we considered nine varieties of tomato diseases they are 1) Bacterial Spot, 2) Early Blight, 3) Late Blight, 4) Leaf mold, 5) Septoria Leaf Spot, 6) Mosaic Virus 7) Yellow Leaf Curl virus 8) Target spot, 9) Spider mites Two-spotted Spider mite.

3.2 Dataset Configuration

By using a prefetch buffer, the dataset can improve its performance. We can yield data from the disk without having I/O blocked using buffered prefetching. During

the initial epoch, Cache () maintains the leaf images in memory after they're loaded off the disk. While the model is training, the Prefetch () function overlaps the data pre-processing and training step model execution. To prefetch the number of batches of images and to get loaded by TensorFlow we use AUTOTUNE feature.

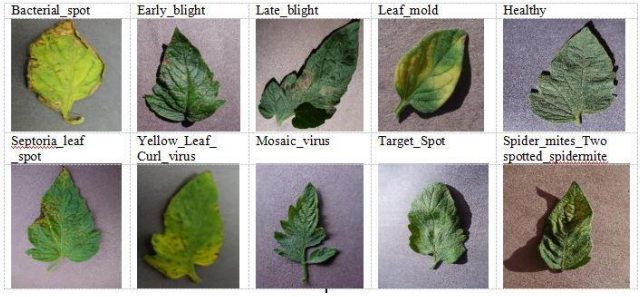


Fig. 2. Class wise sample image from dataset

3.3 Pre-processing of the Dataset

To increase the amount of diversity of data during training, we use the data augmentation. Through applying several transformations like rotation, crop, flip, zoom, contrast, etc new samples will be generated from the existing training samples. This helps the model to expose to various aspects of the data. This is one of the good methods to overcome over-fitting issues and can be applied by adding the more pre-processing layers as a part of the model. In the proposed model, Keras preprocessing layers such as random zoom, random rotation, random flip, etc. are used for preprocessing. Fig.3, depicts the plot of augmented samples for which we have applied the Data Augmentation several times to the same plant image.

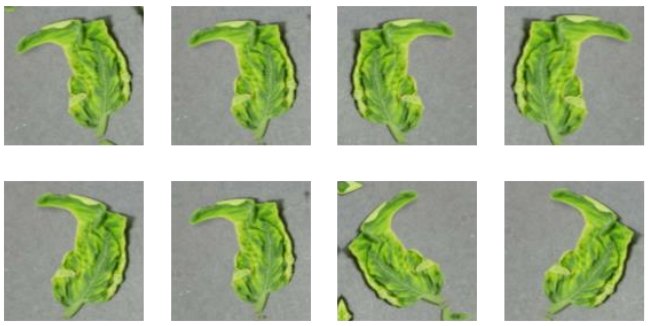


Fig. 3. Data Augmentation was applied to same image several times

3.4 Classification

In this proposed method firstly, CNN architecture is used and whole layers and fil-ters are built from scratch. The dataset we considered has 14529 images belonging to 10 classes. By splitting the dataset into 80:20 ratio for training dataset 11624 images is taken and for the validation dataset 2905 images were taken as per 80:20 ratio split. This model was trained for 50 epochs where images were resized to 180x180 pixels. Furthermore, for feature extraction convolutional and pooling layers are used whereas for classification fully connected layers are used. The purpose of acti-vation layers is to introduce non-linearity into the network Relu, SoftMax (for the last layer) is used. To improve the previous results we have used optimizers like Adam optimizer and sparse categorical loss entropy. The best model used for this project i.e. Model Architecture is shown in Fig. 4. Finally, the model is converted to a TF-Lite model and embedded into a mobile application developed using android studio. Table 1 refers to the details about the model used and Table 2 refers to the model summary.

Table 1

CNN Hyper parameters

Hyper Parameter	Description
Batch size	32
Image size	224
No. of convolution layer	5
No. of max pooling layer	5
No of epochs	50
Activation function	SoftMax, Relu
Optimizer	Adam
Dropout rate	0.2

Table 2

CNN Network Filter

Type	Filter	Output
Convolutional	32	224*224
Maxpooling		112*112
Convolutional	64	112*112
Maxpooling		56*56
Convolutional	64	56*56
Maxpooling		28*28
Convolutional	64	28*28
Maxpooling		14*14
Convolutional	64	14*14
Maxpooling		7*7

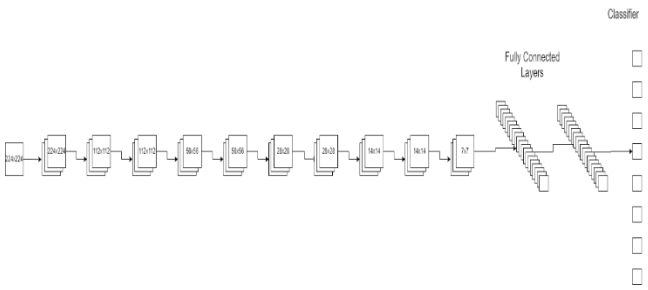


Fig. 4. Model Architecture

3.6 Deployment of the Model

This section refers to the integration of the best CNN model into a mobile application to make use of it in making decisions. The final model is deployed into a mobile application to make sure that farmers and other people can use this app on their smartphones to classify the leaf diseases of the tomato plant. To deliver user-friendly and efficient software that works with or without the internet TF Lite is used. Here the proposed model is converted to a lighter format using TensorFlow Lite due to the complexity and heaviness of CNN models which require lots of memory. Conversion of the CNN model to a TF Lite format doesn't affect accuracy, reduces their file size, and increases execution speed. The inference is known as the process of executing a TF Lite model on a device to make predictions based on the given input data. TF Lite Model can be directly imported to android studio and Java programming language is used for functionality. XML is used for the app interface and after running inference disease class will be predicted. Mobile application is developed because nowadays everyone has a smartphone and it is easy to carry. There is no need for the internet to use this app so as soon as the picture is captured or uploaded the class will be predicted.

3.6.1 Implementation Analysis

The software details of the proposed system are described in this section. The environments used for this project are Android Studio, and Google Colab Pro. Java and Python are programming languages used for Android Studio and Colab.

- *Google Colab Pro*

Python is the programming language, TensorFlow, Matplotlib, NumPy, and OpenCV are the packages used for developing the model. Matplotlib package is utilized for plots, and OpenCV is used for fetching the image from the storage. For Developing the model and for converting the model into a TFLite model, the TensorFlow package is used. To process 14000 images and to train a model Google Colab is not sufficient, So Colab Pro which costs around 10$ (including all taxes) is required as it will provide more Ram, GPU, and storage for working.

- *Android Studio*

XML is used for designing the application interface and for app functionalities Java programming language is used. For certain permissions like camera, storage and location code is written in manifest file.

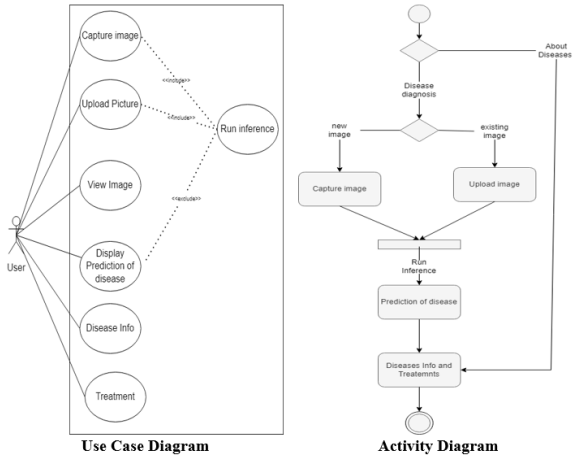


Fig. 4. Use case and activity diagram

In build. Gradle file is required for adding the necessary dependencies like TensorFlow, Firebase etc. TensorFlow Lite model is directly imported to Android Studio. For gradient colors and button style separate xml file is created and applied to the each activity of the app. UML (Unified Modelling Language) diagrams are necessary to visualize a mobile application software. As a part of UML diagrams, Activity Diagram and Use Case Diagram is depicted here. The dynamic behaviour of the system is exhibited by Use Case Diagrams. It discloses the relationships between use cases and the actors. It recognizes the internal as well as external factors and the external view of the system. The concurrent flow of the system is demonstrated by an activity diagram. It is like a flowchart that demonstrates the flow between activities.

4. Experimental Results

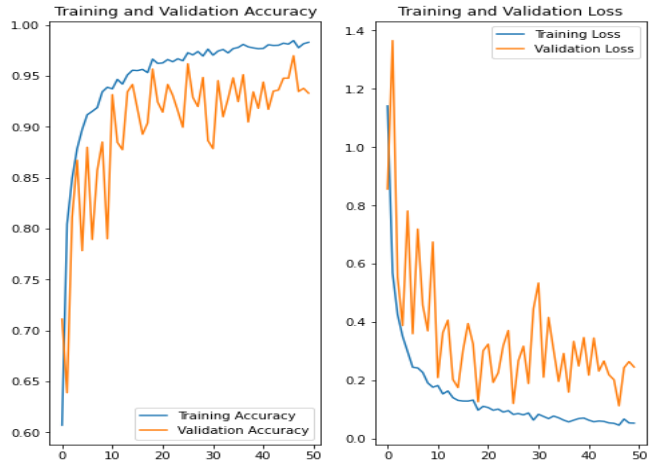
The final model considered is taken based on accuracies, losses of the training dataset, and validation dataset. Initially, the Experiment runs for 30 epochs and there is a constant increase in the accuracy, and a decrease in the loss for both training and validation datasets. To overcome Overfitting Data Augmentation is applied and a dropout layer is added. Along with epochs we experimented with dropout value, no. of layers, and image pixels.

Table 3

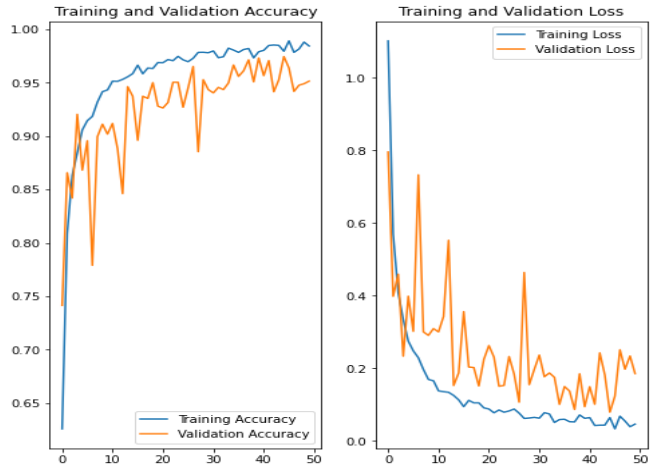
Experimental observation and Analysis

Observations	Pixel	No. of Layers	Dropout rate	Loss	Accuracy	Validation Loss	Validation Accuracy
1	180	6	0.2	0.0803	0.9716	0.2458	0.9329
2	180	6	0.5	0.0462	0.9743	0.1853	0.9515
3	180	10	0.2	0.0642	0.9769	0.0881	0.9759
4	180	10	0.5	0.0826	0.9706	0.1052	0.9656
5	224	10	0.2	0.0397	0.9874	0.0603	0.9797
6	224	10	0.5	0.0682	0.9757	0.0849	0.9766

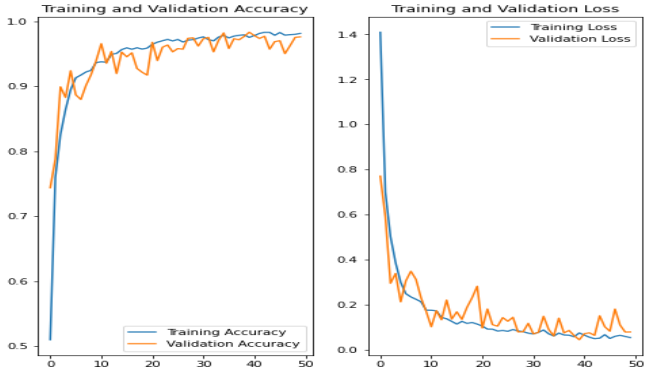
Observations of the experiment are shown in table 3 and their plots are listed in Fig. 5. The accuracy and loss variations of the model were trained using Adam optimizer and Sparse Categorical Cross entropy. Initially, the loss is excessive, and it's far steadily decreased after increasing the number of the epoch.



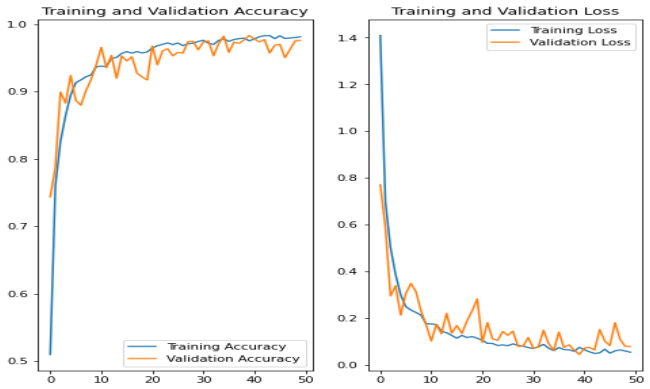
a. Training and validation accuracy with validation loss as 0.24



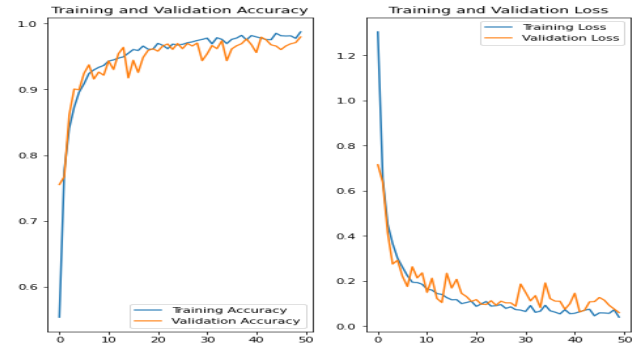
b. Training and validation accuracy with validation loss as 0.18



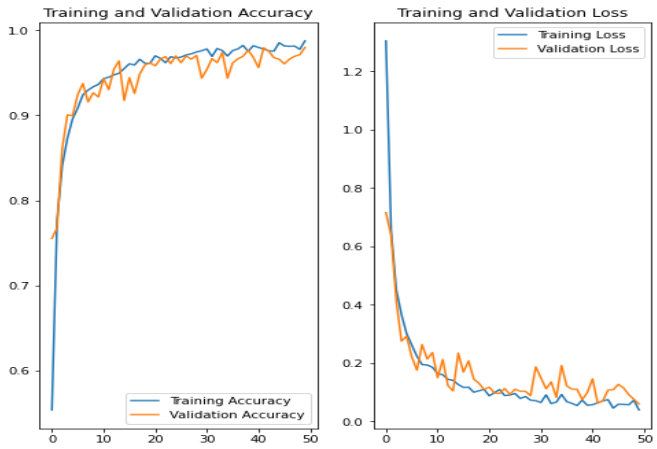
c. Training and validation accuracy with validation loss as 0.08



d. Training and validation accuracy with validation loss as 0.10



e. Training and validation accuracy with validation loss as 0.06



f. Training and validation accuracy with validation loss as 0.08

Fig. 5. Training and validation accuracy with different epochs(parameters are shown in Table 3)

Table 3

Comparison of Tomato Leaf Disease Detection Studies using Deep Learning Techniques.

Study	Pre-trained CNN	Dataset Size	Accuracy
[3]	VGG-16	10,000 images	97.78%
[4]	AlexNet	5,000 images	98.3%
[5]	MobileNet and Inception-v3	15,000 images	98.67%
[6]	VGG-16, Inception-v3, and ResNet-50	4,998 images	97.45%
[7]	GoogLeNet	10,250 images	96.7%
Proposed Work	Custom CNN	14,529 images	98%

The proposed work is a significant contribution to the existing research n automated tomato leaf disease detection using deep learning techniques. As shown in the comparison table, the proposed work achieved an accuracy of 98% using a custom CNN model trained on a dataset of 14,529 tomato leaf images. This accuracy is comparable to the results reported in previous studies that utilized pre-trained CNN models such as VGG-16, AlexNet, MobileNet, Inception-v3, ResNet-50, and GoogLeNet. Furthermore, the proposed work offers additional features such as real-time disease detection, symptom display, and treatment recommendations that are not present in the existing studies. The proposed mobile application can be used offline, making it a suitable tool for farmers and crop cultivators in remote areas. Overall, the proposed work addresses the limitations of existing studies and offers a promising solution for accurate and efficient tomato leaf disease detection. The Fig. 5(e) in the above plots is being selected to extend the performance evaluation. The selected final model is converted into a TF Lite model format and embedded into a mobile application using Android Studio. With the user-friendly app design, users can easily under-stand how to use this application. Signup Activity is for user Registration, login activity for user login. A home screen contains the camera icon which is used for capturing the disease picture, file icon is to choose the image from the app. We get an immediate classified result along with confidence percentage as soon as you upload the picture as shown in Fig. 8. We can also get the information like symptoms and treatments related to the disease. Setting feature has many sub features like Edit profile, Manage Accounts, Logout options are for personal information management, About, Privacy Policy, Terms and Conditions are to know more about app.

4. Conclusion and Future Work

A mobile application for real-time tomato leaf disease detection is the final system presented by this project. For achieving this system, firstly CNN model is trained with 14529 tomato leaf images and well optimized with decent accuracy for mobile deployment. The performance of the model is evaluated based on the training accuracy, validation accuracy of around 98%, and validation loss parameter. This developed application will almost precisely classify the leaf diseases of the tomato plant and after detection symptoms, treatments are also displayed. Various organic and chemical controls are displayed to treat the disease. For chemical control, various products are suggested according to the toxic level and along with components in it. Farmers and other crop cultivators can use this application without the internet also so this application can be used in remote areas also. The future scope of this project is to continuously improve the existing model by adding more variety of diseases and other crop diseases. Add the feature of location access, we can add voice support which will translate the text to the local language speaking at that place. According to the location accessed, we can further add beneficial features such as suggesting the next seasonal crop, productivity improvement, pesticides, and their effects, call support from nearby agriculture officers, etc. By contacting the local store owners, we can add a purchase support, so that farmers can buy the required products from the local store or nearby store.

5. References

[1] M. Agarwal, A. Singh, S. Arjaria, A. Sinha, and S. Gupta, “ToLeD: tomato leaf disease detection using convolution neural network”, Procedia Computer Science, pp. 293-301, 2020.

[2] M. Brahimi, K. Boukhalfa and A. Moussaoui, “Deep learning for tomato diseases: classification and symptoms visualization, Applied Artificial Intelligence, 31:4, 299-315, DOI: 10.1080/08839514.2017.1315516

[3] E. Fujita, et al., "Basic investigation on a robust and practical plant diagnostic system", 15th IEEE international conference on machine learning and applications, IEEE, 2016.

[4] U. Shruthi, V. Nagaveni, and B. K. Raghavendra. "A review on machine learning classification techniques for plant disease detection", 5th International conference on advanced computing and communication systems, IEEE, 2019.

[5] M. H. Saleem, J. Potgieter, and K. M. Arif, "Plant disease detection and classification by deep learning", Plants 8.11: 468, 2019.

[6] J. Trivedi, Y. Shamnani, and R. Gajjar, "Plant leaf disease detection using machine learning", International conference on emerging technology trends in electronics communication and networking. Springer, Singapore, 2020.

[7] S. Adhikari, B. Shrestha, B. Baiju, and S. Kumar, “Tomato plant diseases detection system using image processing”, 1st KEC Conference on Engineering and Technology, Lalitpur vol. 1, pp. 81-86, 2018.

[8] L. Loyani, and D. Machuve, “a deep learning-based mobile application for segment-ing tuta absoluta’s damage on tomato plants”, Engineering, Technology and Applied Science Research, pp. 7730-7737, 2021.

[9] M. Hasan, B. Tanawala, and K. J. Patel, “Deep learning precision farming: tomato leaf disease detection by transfer learning”, Proceedings of 2nd International Conference on Advanced Computing and Software Engineering, 2019.

[10] P. Tm, A. Pranathi, K. SaiAshritha, N. B. Chittaragi, and S. G. Koolagudi, “Tomato leaf disease detection using convolutional neural networks”, 11th International conference on contemporary computing, IEEE, pp. 1-5, 2018.

[11] T. Anandhakrishnan, and J. S. Murugaiyan, “Identification of tomato leaf disease detection using pretrained deep convolutional neural network models”, Scalable Compu-ting: Practice and Experience. 21(4), pp.625-635, 2020.

[12] R. Thangaraj, S. Anandamurugan, and Kaliappan, “Automated tomato leaf disease classification using transfer learning-based deep convolution neural network”, Journal of Plant Diseases and Protection, V.K 128(1), pp.73-86, 2021.