



Android Application for Tomato Leaf Disease Prediction Based on MobileNet Fine-tuning

Mutia Fadhillah¹, Des Suryani²

^{1,2}Department of Informatics, Faculty of Engineering, University of Islam Riau, Pekanbaru, Indonesia

¹tiafadhillah@eng.uir.ac.id, ²des.suryani@eng.uir.ac.id

Abstract

Tomato is one of the most well-known and widely cultivated plants in the world. Tomato production result is affected by the conditions of the plants when they are cultivated. It may decrease due to leaf plant disease caused by climate change, pollinator decrease, microbial pests, or parasites. To prevent this, an image-based application is needed to identify tomato plant disease based on visually unique patterns or marks seen on leaves. In this paper, we proposed a CNN fine-tuned model that is based on MobileNet architectures to identify tomato leaf disease for mobile applications. Based on the results tested by K-fold cross-validation, the best accuracy achieved by the proposed model is 97.1%. In addition, the best average precision, recall, and F1 Score are 99.8%, 99.8%, and 99.5% respectively. The model with have best results is also implemented into Android-based mobile applications.

Keywords: deep learning; computer vision; android application; tomato leaf disease

1. Introduction

Tomato is one of the most widely cultivated plants in the world alongside corn, potato, wheat, etc. Tomato is also one of the well-known crops and can be found as a cooking ingredient in the kitchen with a wide variety of types of cuisine. It is probably because tomato has many nutrients that are beneficial to the body such as vitamin C, potassium, and folate. These vitamins and minerals are useful as antioxidants, anti-inflammatory, and anti-cancer[1].

The results of agricultural production, including tomatoes, are strongly influenced by the conditions of the plants when they are cultivated. The quantity and quality may decrease if the plants are infected by disease. Plants that are infected with the disease have obvious marks or lesions on plant parts such as leaves. Each type of disease has a visually unique pattern that can be distinguished and diagnosed [2]. Traditionally, agricultural experts usually have on-site observations to identify plant diseases. It has several limitations, such as being time-consuming, tiresome, and less efficient. In addition, there is the possibility of misjudgment due to weariness and lack of experience.

Nowadays, with recent developments in various agricultural technologies, it is possible to leaf plant disease automatically using a computer vision or deep

learning approach. So far, some studies are related to plant leaf disease identification and detection. [3] proposed method to identify the leaf spot using image processing techniques which are image segmentation by k-means clustering and feature extractions, then used Artificial Neural Network for classification. [4] proposed deep learning model which is a Convolutional Neural Network (CNN) for the detection and classification of plant leaf disease. [5] compared some machine learning and deep learning techniques to identify citrus plant disease.

Meanwhile, studies that focus on tomato leaf disease detection have been conducted in recent years. Some studies proposed a CNN-based model to detect leaf diseases. [6] proposed comparison of some CNN fine-tuned models (LeNet, VGG, ResNet, Xception) with evaluation of two types of dataset variants which are colour images and segmented images. [7] proposed CNN model with three convolutional and max-pooling layers followed by one fully connected layer and output layer. [8] proposed CNN model by reconstructing the Deep Residual Dense Network to identify tomato leaf disease.

Then, [9] and [10] modify a CNN-based model by adding an attention module to the proposed method. [11] the proposed method of tomato image augmentation using Deep Convolutional Generative

Adversarial Network (DCGAN). Augmented images and original images are used as input data to the classification model. [12] proposed method is divided into some parts which are image denoising and enhancement using Binary Wavelet Transform combined with Retinex, image object separation from the background using Artificial Bee Colony algorithm, and disease detection using Both-channel Residual Attention Network mode. But most related studies above are developed for running on computers with powerful specifications and resources. [13] adopted CNN model that can be implemented for smart mobile applications. However, it has not developed into a mobile application that can be used on mobile devices.

Based on previous studies and the problems mentioned, the prediction of leaf disease is needed for the early prediction of disease in plants. It is necessary to prevent a decrease in the production of fruit or vegetable crops such as tomatoes. Apart from that, the prediction model needs to be implemented into an application that can be used by farmers or general users more easily and efficiently in the future. Hence, we proposed a deep learning model that adapted from the best existing CNN model for mobile application, MobileNet which used a fine-tuning approach for training the model. In this study, we modified MobileNet architecture by changing and simplifying the fully connected layer into one classifier layer that has ten nodes representing the number of classes in tomato leaf diseases. In addition, K-Fold Cross Validation is used for evaluating the proposed method's performance. It results in a high accuracy score which is up to 97%.

2. Research Methods

The research processes are shown in Figure 1. The first process is collecting the dataset and then separating it into three parts: training data, validation data, and testing data. Next, training the proposed model and evaluating it using K-Fold Cross Validation. The model from the folding step in K-Fold cross-validation that has the best accuracy score is then selected as the best model. The final process is building an Android application for tomato leaf disease prediction using the best model from the evaluation process.

2.1 Dataset Collection and Preprocessing

The tomato leaf disease pictures are obtained from the dataset published in [14]. The dataset consists of 39 different classes of plant leaves such as apple leaf, blueberry leaf, corn leaf, and tomato leaf which contain 61,486 images. It uses six different augmentation techniques (image flipping, Gamma correction, noise injection, PCA colour augmentation, rotation, and Scaling) to increase the data-set size.

In this study, we collected a subset of the plant leaf image dataset. It contains 5000 images chosen

randomly that belong to ten categories that related to tomato leaf and each category consists of 500 images. The categories are Bacterial Spot, Early Blight, Healthy Leaf, Late Blight, Leaf Mold, Septoria Leaf Spot, Spider Mites, Target Spot, Tomato Mosaic Virus, and Tomato Yellow Leaf Curl Virus. Then, the dataset of tomato leaf images is divided into three data partitions, 1000 images in the testing dataset, and 4000 images in the training and validation dataset. Example images are shown in Figure 2. Meanwhile, for the image preprocessing, the size of all images is set to 128×128 and the format is jpeg.

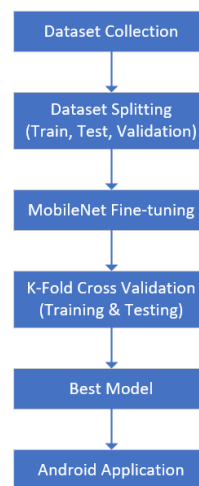


Figure 1. Research Processes

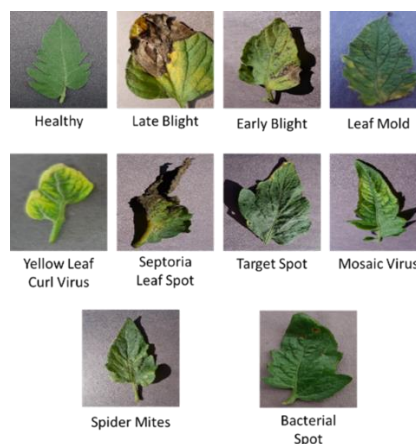


Figure 2. Example of Tomato Leaf Images

2.2 MobileNet Fine-tuning

Conventional machine learning approaches make predictions based on statistical data and the prediction model is trained based on labelled data. Meanwhile, deep learning is a subset of machine learning that requires a large dataset and a lot of computational power due to the process of training deep neural networks. The transfer learning approach can be used to solve this problem. It has benefits in terms of time complexity and large required datasets [15].

Transfer learning is a new task-learning process through knowledge transfer from the deep learning model that has related tasks and has already been learned. The most popular approach in transfer learning is copying a trained base network to the first layers of the target network, and then adding new layers for specific target tasks. We can choose which part of the target network needs to be trained and update the weights. This approach is known as fine-tuning. In this study, a pre-trained MobileNet model is used as the base layer in the proposed model. MobileNet is one of the efficient models for mobile and embedded vision applications [16]. It is based on a streamlined architecture that uses depth-wise separable convolutions.

Figure 3 shows the network architecture of the proposed model. The input data is an RGB image that has been resized in preprocessing, with a size of 128×128 . The next part is CNN Base Architectures which is MobileNet. It consists of one convolutional layer that uses 32 filters with kernel size 3×3 , followed by some Depth-wise Separable (DS) Convolutional layers that have various batch sizes and number of filters for each layer, and the last layer is the average pool layer. DS Convolutional layer is a sequential model containing some layers which consist of depthwise and pointwise convolution layers. Both convolution layers are followed by the batch normalization layer and ReLU activation function.

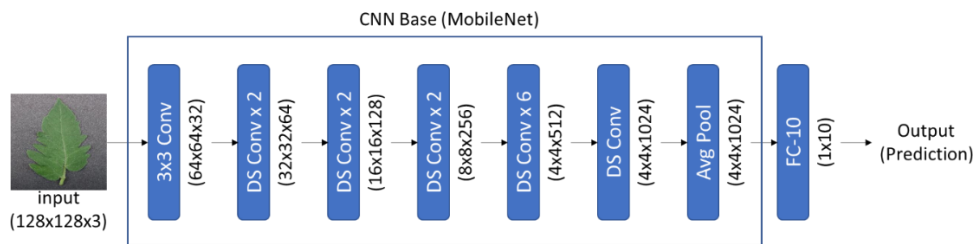


Figure 3. MobileNet Fine-tuning Architecture

Figure 4 shows a visualization of the DS Convolutional Layer. The fine-tuning process in this study is conducted by adding one fully connected layer at the end of the model which is also the output layer of the proposed model. The output layer uses the Softmax activation function to get a vector of probabilities that contains probability distribution over the classes.

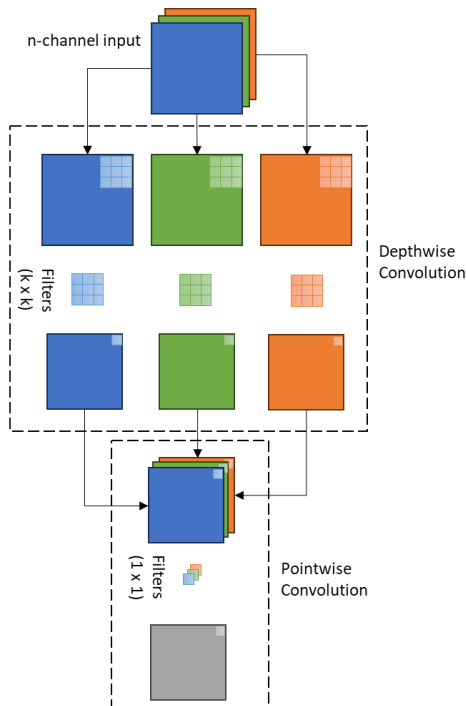


Figure 4. Depthwise Separable Convolutional Layer

In this case, it is a multiclass classification with ten categories, so the Softmax activation returns an output vector which has ten probability values that correspond with each category.

In this study, the entire weight of the proposed network architecture will be updated during the training phase. The goal of the training phase is to maximize prediction accuracy with an iterative process until the loss function is minimized. To predict the tomato leaf disease category, cross-entropy is used as a loss function. Its formula is described in Formula 1.

$$CE = - \sum_{i=1}^n g_i \log p_i \quad (1)$$

n is the number of categories, g_i is the ground truth or actual label and p_i is the predicted label.

2.3. K-Fold Cross Validation

Cross-validation is a statistical method to evaluate and compare machine learning algorithms by dividing data into two parts: one used to train a model and the other used to validate or test the model [17]. It is commonly used to estimate trained model skills on unseen data. It is also used to avoid overfitting during the training phase.

K-Fold Cross Validation is a basic form of Cross-Validation. In this form, the dataset is divided into k equally sized folds and runs the training and testing process iteratively. For each running step, one fold is considered as testing data and the other folds as training data. In the next running steps, will be conducted similar process with a different fold for testing data. Figure 5 shows the partition data for each running

process using K-Fold Cross Validation with K=3. In this study, we used K-Fold cross-validation with K=5. Fold-1 contains 100 first images on each class for the testing set which has a total of 1000 images and other images for the training set. Fold-2 contains the next 100 images for the testing set the rest images for the training set, and so on for the next folds.

1	Test	1	Train	1	Train
2	Train	2	Test	2	Train
3	Train	3	Train	3	Test

Figure 5. Data Partition Example for K=3

During the iterative process in K-Fold Cross Validation, some measurements are used to evaluate the performance of the proposed model including accuracy, precision, recall, and F1 score. Accuracy is used to calculate how close the prediction result category and actual category is. Precision is calculated to know how many of the certain category predictions made are correct. Recall is measured to know how many of the certain category data is correctly predicted. For multi-class classification, the precision and recall of each class can be measured using Formula 2 and 3.

$$P_C = \frac{TP_C}{N.of\ Data\ Predicted\ as\ C} \quad (2)$$

$$R_C = \frac{TP_C}{N.of\ Data\ Labeled\ as\ C} \quad (3)$$

P_C and R_C are precision and recall of class C . TP_C (True Positive) is several data that are correctly predicted as class C . P_C is the ratio between TP_C and several data predicted as class C and R_C is the ratio between TP_C and several data labeled as class C . Furthermore, the F1 score is calculated by Formula 4.

$$F1_C = 2 * \frac{P_C * R_C}{P_C + R_C} \quad (4)$$

$F1_C$ is the F1 score of class C , P_C is the Precision score of class C , and R_C is the Recall score of class C .

2.4 Android Application

The last process of this research is implementing the best model collected from the training and evaluation process into a mobile application. One of the widely used operating systems on mobile devices is Android. In this study, the Android application will be deployed using Android Studio IDE and Tensorflow Lite library. TensorFlow Lite is an open-source library that is published by Google TensorFlow. It is a set of tools that enables on-device machines to run the models on mobile, embedded, and edge devices[18].

The best model of MobileNet fine-tuned architectures is selected by comparing accuracy results in K-Fold Cross Validation. The trained model resulting from the folding process that has the highest accuracy will be saved as the best model. The saved model is converted into a Tensorflow Lite (tflite) model using a converter

that is provided in the library. Then, the model, converted to file format, is implemented into an Android application. This process is shown in Figure 6.

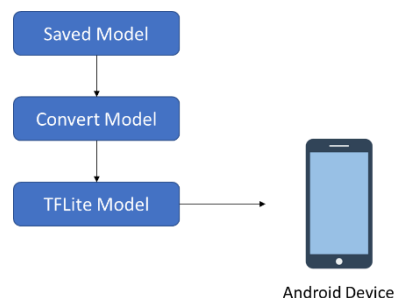


Figure 6. Model Conversion Using TensorFlow Lite

3. Results and Discussions

The results of this study are the testing results from K-Fold Cross Validation with K=5. In addition, some hyperparameters are set during the training phase which are 32 batch size, 100 epoch, adam optimizer and 0.001 dropout. The training and testing phases were executed on each fold using some performance measurements to evaluate the proposed CNN model. The measurements are accuracy for each fold, precision, recall, and F1 score for each class label corresponding with each fold. The tables of the entire evaluation results use a number-based label that represents the tomato leaf category. The list of class labels and category names respectively is shown in Table 1.

Table 1. Class Label and Category Name

Class Label	Category
1	Bacterial Spot
2	Early Blight
3	Healthy Leaf
4	Late Blight
5	Leaf Mold
6	Septoria Leaf Spot
7	Spider Mites
8	Target Spot
9	Tomato Mosaic Virus
10	Tomato Yellow Leaf Curl Virus

3.1 Performance Evaluation

Table 2 shows the accuracy results of tomato leaf disease prediction based on K-Fold Cross Validation with size K=5. Based on the results, the proposed method has a very high accuracy for predicting tomato leaf disease. As shown in Table 2, the highest accuracy score is the proposed model on the first fold, which is 97.10%. Meanwhile, the average accuracy is 94.42%.

Table 2. Accuracy (%) Results on 5-Fold Cross Validation

Fold	Accuracy (%)
1	97.10
2	97.00
3	95.60
4	95.80
5	96.70
Average	94.42

The quality of the performance of the model is not only measured by accuracy but can also be seen by precision and recall scores. The precision results of the proposed model are shown in Table 3. It contains a precision score of each class that corresponds to each fold testing phase, then calculates the average for a final precision score, namely Average Precision. Based on the result shown in Table 3, all categories have an average precision above 90%. It means the proposed model also has a great ability to predict each class precisely where the mean of average precision is 96.53%.

In addition, the best average precision is the Healthy Leaf category which has a 99.80% average precision score. Next, the second-best score is the Tomato Mosaic Virus category which is 98.07%, followed by Leaf Mold, Spider Mites, and Bacterial Spot which have 98.03%, 97.65%, and 97.64% precision scores respectively. Furthermore, the worst score is obtained by the Early Blight category which has 92.93% in average precision.

Table 3. Precision (%) Results on 5-Fold Cross Validation

Class	Fold					Avg (%)
	1	2	3	4	5	
1	97.09	100	98.91	97.91	94.29	97.64
2	92.08	98.92	93.81	88.67	91.18	92.93
3	100	98.98	100	100	100	99.80
4	100	100	85.71	97.84	92.78	95.27
5	100.	96.00	95.19	100.00	98.97	98.03
6	92.6	88.99	100	90.56	94.06	93.24
7	98.93	100	93.33	97.95	98.04	97.65
8	96.90	92.45	98.92	91.42	96.96	95.33
9	96.15	96.15	100	98.04	100	98.07
10	97.03	100	92.52	97.09	100	97.33
Mean Average Precision						96.53

Table 4. Recall (%) Results on 5-Fold Cross Validation

Class	Fold					Avg (%)
	1	2	3	4	5	
1	100	96	91	94	99	96
2	93	92	91	94	93	92.6
3	100	98	98	100	100	99.2
4	93	97	96	91	90	93.4
5	99	96	99	91	97	96.4
6	100	97	93	96	95	96.2
7	93	98	98	96	100	97
8	94	98	92	96	96	95.2
9	100	100	99	100	100	99.8
10	98	98	99	100	97	98.4

Meanwhile, Table 4 shows the recall results on 5-Fold Cross Validation. It contains similar format results with precision, where each class has an average recall that is calculated from all scores in each folding process on cross-validation. As shown in Table 4, the average recall score reaches 99.8% with the best result obtained by the Tomato Mosaic Virus category. Then, the next top best scores are obtained by the Healthy Leaf, Tomato Yellow Leaf Curl Virus, and Spider Mites category which has 99.20%, 98.40%, and 87% scores in average precision respectively. Meanwhile, the lowest

score of average recall is 92.6% which belongs to the Early Blight category.

Table 5. F1 Score (%) Results on 5-Fold Cross Validation

Class	Fold					Avg (%)
	1	2	3	4	5	
1	98.52	97.96	94.79	95.92	96.59	96.76
2	92.54	95.33	92.38	91.26	92.08	92.72
3	100	98.49	98.99	100	100	99.50
4	96.37	98.48	90.56	94.30	91.37	94.22
5	99.50	96	97.06	95.29	97.98	97.16
6	96.16	92.82	96.37	93.20	94.53	94.62
7	95.87	98.99	95.61	96.97	99.01	97.29
8	95.43	95.14	95.33	93.65	96.48	95.21
9	98.04	98.04	99.50	99.01	100	98.92
10	97.51	98.99	95.65	98.52	98.48	97.83

The last measurement for model evaluation is the F1 score. Table 5 shows the results of the F1 score based on 5-fold cross-validation. F1 score results depend on the precision and recall results of the proposed model. Based on the results shown in Table 5, the best F1 score is 99.5% which belongs to the Healthy Leaf category. The next three best results are obtained by the Tomato Mosaic Virus with a 98.92% F1 score, followed by the Tomato Yellow Leaf Curl Virus category with a 97.83% F1 score and the Spider Mites category that has 92.29% F1 score.

Furthermore, based on precision, recall, and F1 score shown in Table 3, Table 4, and Table 5, the Healthy Leaf and Tomato Mosaic Virus category have better results compared with other categories. They have above 99% scores in average precision, average recall, and F1-Score. Meanwhile, Early Blight has the worst score of these three measurements, but it still has great performance which has above 92% precision, recall, and F1 score.

Based on the results of performance analysis using several calculations, namely accuracy, precision, recall, and f-1 score, it can be concluded that the proposed model has great performance in predicting tomato leaf disease. The proposed model has an accuracy rate of up to 97.1%, a mean average precision is 96.53%, a recall precision is up to 99.8%, and an f-1 score is up to 99.5%.

To evaluate the performance of the proposed method, we also conduct a result comparison analysis with previous studies related to tomato leaf disease identification and detection. We use the best accuracy and average accuracy score to compare the performance of the proposed method with previous related studies. The accuracy comparison is shown in Table 6.

Based on the best accuracy obtained in the testing phase, the proposed method has better performance than almost all previous related studies, except the attention-based network architecture proposed in [9], [10]. They have 98% and 99.24% accuracy scores, meanwhile, our proposed study has a 97.1% accuracy

score. But, compared with other previous related studies, our proposed method has higher accuracy, whereas other studies have an accuracy score below 96%.

Table 6 Accuracy Comparison Between Related Studies and Proposed Method

Method	Accuracy (%)
DCGAN-Based [11]	94.33%
B-ARNet [12]	89%
Attention-Based [9]	98%
Attention-Based [10]	99.24%
Residual Dense Network [8]	95%
ToLED [7]	91.2%
Smart Mobile – CNN [13]	90.3%
Proposed Method	
Best Accuracy	97.1%
Average Accuracy	94.42%

Furthermore, based on the average accuracy of our proposed method, it has better performance than some previous studies. They are B-ARNet, ToLED, and mobile CNN architecture proposed by [7], [12], [13] respectively which have accuracy scores below 90%. Besides that, our proposed method has a slightly better accuracy score compared to the DCGAN-based network architecture proposed in [11] which is only 0.09% below our proposed average accuracy score. Meanwhile, other previous studies, which are attention-based and deep residual dense networks proposed in [8]–[10], have better accuracy scores compared to our proposed method where the accuracy score is 95% and above. But, compared with previous studies that also proposed a method for mobile applications [13], our proposed method has better performance than that related study. It only has a 90.3% accuracy score while our proposed method has a 94.42% average accuracy score and a 97.1% best accuracy score.

In addition, some examples of prediction results for each fold in the testing phase are shown in Table 7-11. Each table contains one best and worst example of prediction with confidence value. For the worst-case example, it also contains confidence value to predict its actual category. Tables 7 and 8 show prediction results on folds 1 and 2, the best examples are the categories that had the best overall result in measurements which are Healthy Leaf and Tomato Mosaic Virus.

The worst example is the Early Blight category which has the worst performance results. Based on Table 7, the first image is a Tomato Mosaic Virus image, and the proposed method successfully predicted it as a Tomato Mosaic Virus with a 99.9% confidence rate. The second image is Early Blight class, and the proposed method incorrectly predicted as Septoria Leaf Spot with 54.42% confidence rate, meanwhile, the proposed method only has 25.77% confidence that the image is Early Blight class. Meanwhile, Table 9-11 shows the prediction results of folds 3, 4, and 5 respectively.

The best and worst example images were collected randomly for each fold besides the categories that have already been in Tables 7 and 8 as image examples.

Table 7. Examples of Prediction Results of Fold-1



Image and Category	Prediction Result
 Tomato Mosaic Virus	Tomato Mosaic Virus Confidence: 99.99%
 Early Blight	Septoria Leaf Spot Confidence: 54.42%
	Early Blight Confidence: 25.77%

Table 8. Examples of Prediction Results of Fold-2



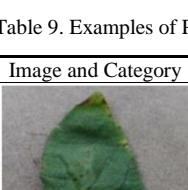
Image and Category	Prediction Result
 Healthy Leaf	Healthy Leaf Confidence: 100%
 Target Spot	Target Spot Confidence: 92.77%
 Early Blight	Early Blight Confidence: 6.31%

Table 9. Examples of Prediction Results of Fold-3




Image and Category	Prediction Result
 Bacterial Spot	Bacterial Spot Confidence: 99.99%
 Late Blight	Late Blight Confidence: 42.72%
 Bacterial Spot	Bacterial Spot Confidence: 41.90%

Table 10. Examples of Prediction Results of Fold-4





Image and Category	Prediction Result
 Leaf Mold	Leaf Mold Confidence: 100%
 Leaf Mold	Early Blight Confidence: 47.69%
	Leaf Mold Confidence: 7.3%

Table 11 Examples of Prediction Result of Fold-5

Image and Category	Prediction Result
 Septoria Leaf Spot	Septoria Leaf Spot Confidence: 98.72%
 Septoria Leaf Spot	Leaf Mold Confidence: 99.52%
	Septoria Leaf Spot Confidence: 0.4%

3.2 Android Application Result

The best model obtained from the testing phase using K-Fold Cross Validation is saved for developing an Android-based mobile application. Based on the accuracy results in Table 2, the best score is the model trained on fold 1. It is then converted to a TensorFlow Lite model that can be implemented into an Android application. The application is developed using Android Studio IDE, and an Android mobile device, Samsung S20, for running the application. The application user interface can be seen in Figure 7. As shown in Figure 7, the user can input an image from the camera or select it from a mobile phone gallery. The input image will be shown in the user interface along with the prediction result and its confidence value of prediction. Figure 7 shows two best cases of prediction tomato leaf disease using an Android application.

Meanwhile, for the worst-case example of tomato leaf disease prediction using the application shown in Figure 8. The left image shows the worst-case result based on a proposed method that was implemented in the application. It shows wrong prediction results for

predicting tomato leaf disease. The right image shows the limitation of the application where it is not capable of differentiating leaf images from other objects. As shown in Figure 8, the right image is an example of tomato leaf disease prediction using a cat image. It predicts it as a Late Blight category with 99.98% confidence. The application needs a lot of improvement for future research for this case. The application also can't predict using a live camera. For this case, it certainly requires further research in the development of models for live camera prediction and detection. Apart from that, this research is still limited to developing mobile applications that are only based on Android, not multi-platform. So, the application cannot run on smartphones other than Android. It also can be a consideration for further research to develop a multi-platform mobile application for predicting tomato leaf disease.

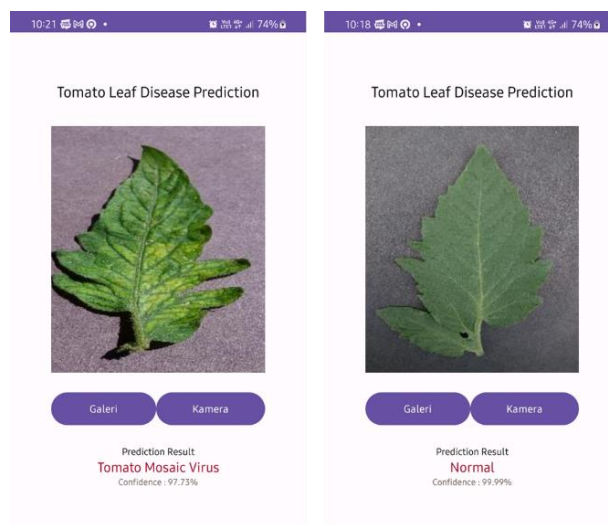


Figure 7 Android Application for Tomato Leaf Disease Prediction (Best-case Example)

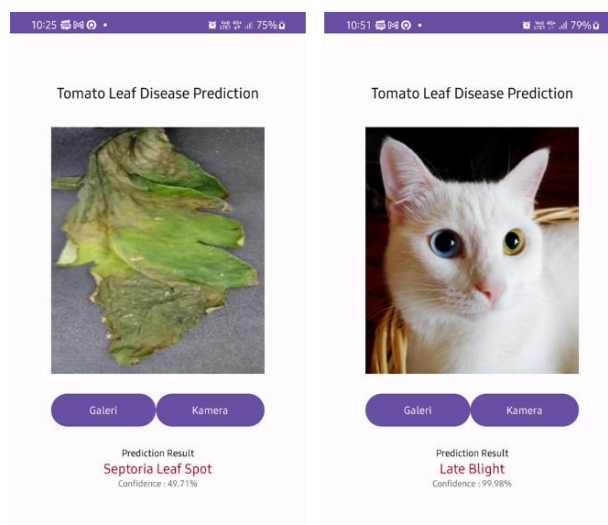


Figure 8 Android Application for Tomato Leaf Disease Prediction (Worst-case Example)

4. Conclusion

In this study, we proposed a model for tomato leaf disease prediction based on fine-tuning MobileNet architecture. It conducts the training and testing phase using k-fold cross-validation with k=5. Based on the performance evaluation results, our proposed model has great performance where the highest accuracy is 97.1% obtained from the first fold in k-fold validation. Based on the best accuracy score, our proposal achieved better performance than some of the previous related studies. Meanwhile, our proposed method has achieved 99.8% for the precision and recall score, and 99.5% for the F1 score. The best model obtained from testing is also implemented into the application. It was developed into an Android-based mobile application. Despite some limitations of the application, it is capable of doing prediction tasks for tomato leaf disease. The application certainly needs some improvements in future studies, such as a real-time detection feature for the application and multi-platform deployment.

Reference

- [1] B. Salehi *et al.*, "Beneficial effects and potential risks of tomato consumption for human health: An overview," *Nutrition*, vol. 62, 2019. doi: 10.1016/j.nut.2019.01.012.
- [2] L. Li, S. Zhang, and B. Wang, "Plant Disease Detection and Classification by Deep Learning - A Review," *IEEE Access*, vol. 9, 2021. doi: 10.1109/ACCESS.2021.3069646.
- [3] C. Usha Kumari, S. Jeevan Prasad, and G. Mounika, "Leaf disease detection: Feature extraction with k-means clustering and classification with ANN," in *Proceedings of the 3rd International Conference on Computing Methodologies and Communication, ICCMC 2019*, 2019. doi: 10.1109/ICCMC.2019.8819750.
- [4] H. S. Nagamani and H. Sarojadevi, "Tomato Leaf Disease Detection using Deep Learning Techniques," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 1, 2022, doi: 10.14569/IJACSA.2022.0130138.
- [5] R. Sujatha, J. M. Chatterjee, N. Z. Jhanjhi, and S. N. Brohi, "Performance of deep learning vs machine learning in plant leaf disease detection," *Microprocess Microsyst*, vol. 80, 2021, doi: 10.1016/j.micpro.2020.103615.
- [6] A. Kumar and M. Vani, "Image-Based Tomato Leaf Disease Detection," in *2019 10th International Conference on Computing, Communication and Networking Technologies, ICCCNT 2019*, 2019. doi: 10.1109/ICCCNT45670.2019.8944692.
- [7] M. Agarwal, A. Singh, S. Arjaria, A. Sinha, and S. Gupta, "ToLeD: Tomato Leaf Disease Detection using Convolution Neural Network," in *Procedia Computer Science*, 2020. doi: 10.1016/j.procs.2020.03.225.
- [8] C. Zhou, S. Zhou, J. Xing, and J. Song, "Tomato Leaf Disease Identification by Restructured Deep Residual Dense Network," *IEEE Access*, vol. 9, 2021, doi: 10.1109/ACCESS.2021.3058947.
- [9] R. Karthik, M. Hariharan, S. Anand, P. Mathikshara, A. Johnson, and R. Menaka, "Attention embedded residual CNN for disease detection in tomato leaves," *Applied Soft Computing Journal*, vol. 86, 2020, doi: 10.1016/j.asoc.2019.105933.
- [10] S. Zhao, Y. Peng, J. Liu, and S. Wu, "Tomato leaf disease diagnosis based on improved convolution neural network by attention module," *Agriculture (Switzerland)*, vol. 11, no. 7, 2021, doi: 10.3390/agriculture11070651.
- [11] Q. Wu, Y. Chen, and J. Meng, "DCGAN-Based Data Augmentation for Tomato Leaf Disease Identification," *IEEE Access*, vol. 8, pp. 98716–98728, 2020, doi: 10.1109/ACCESS.2020.2997001.
- [12] X. Chen, G. Zhou, A. Chen, J. Yi, W. Zhang, and Y. Hu, "Identification of tomato leaf diseases based on combination of ABCK-BWTR and B-ARNet," *Comput Electron Agric*, vol. 178, Nov. 2020, doi: 10.1016/j.compag.2020.105730.
- [13] A. Elhassouny and F. Smarandache, "Smart mobile application to recognize tomato leaf diseases using Convolutional Neural Networks," in *Proceedings of 2019 International Conference of Computer Science and Renewable Energies, ICCSRE 2019*, 2019. doi: 10.1109/ICCSRE.2019.8807737.
- [14] A. P. J and G. GOPAL, "Data for Identification of Plant Leaf Diseases Using a 9-layer Deep Convolutional Neural Network," *Mendeley Data*, VI, 2019, doi: 10.17632/tywbtsjrjv.1.
- [15] G. Vrbančić and V. Podgorelec, "Transfer learning with adaptive fine-tuning," *IEEE Access*, vol. 8, 2020, doi: 10.1109/ACCESS.2020.3034343.
- [16] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv preprint arXiv:1704.04861*, 2017, doi: https://doi.org/10.48550/arXiv.1704.04861.
- [17] P. Refaeilzadeh, L. Tang, and H. Liu, "Cross-Validation," in *Encyclopedia of Database Systems*, New York: Springer, 2016. doi https://doi.org/10.1007/978-1-4899-7993-3_565-2.
- [18] "Tensorflow Lite." <https://www.tensorflow.org/lite/guide> (accessed Apr. 28, 2023).