Evaluation of The Deep Learning Techniques to Identify Plant Diseases Using Leaf Images

Harry Yuliansyah[1], Rudy Hartanto[2], Indah Soesanti[3]

[1.2.3]Department of Electrical Engineering and Information Technology, Universitas Gadjah Mada, Yogyakarta, Indonesia [1]Electrical Engineering, Insitut Teknologi Sumatera, Lampung, Indonesia
indahsoesanti@ugm.ac.id

Abstract: Successful farming is influenced by various techniques conducted by farmers in identifying the types of diseases that affect plant yields to avoid greater losses. Therefore, this study aims to evaluate the deep learning techniques in identifying plant diseases using leaf images. Furthermore, an artificial intelligence approach was used to identify types of plant diseases. During the deep learning training, about 11 deep learning architectural models and consisting of 38 classes in the dataset were used. The results showed that the highest minimum accuracy value obtained was 87.10%, with only one class having an accuracy value below 90%.
Keywords: plant disease, deep learning, leaf image, agriculture, artificial intelligence

1. Introduction

Indonesia is an agricultural country where farming is the primary source of income for most people. According to the Central Bureau of Statistics (BPS), the number of workers in the agricultural sector was about 35.7 million people in 2018, which accounted for 28.79% of the total working population of about 124.01 million. Furthermore, this number decreased by 121.02 million people compared to the previous year, when it was 35.9 million, accounting for 29.68% of the total working population.

Subsequently, farmers face a significant challenge in combating diseases, which can spread to other plants if not treated quickly. Therefore, the rate at which these diseases are identified is an important factor that contributes to the success of agriculture. Also, an accurate treatment plan in response to the various plant diseases can prevent farmers from incurring greater losses.

The diseases symptom presented by the plant can be used by the farmer to identify various types of diseases [1]. Another method for identifying various plant diseases involves the use of digital image processing, which uses spectroscopic techniques, such as visible light, infrared, and multispectral bands. Furthermore, this method does not require any special user skills since it has a low level of accuracy in recognizing plant disease patterns.

Therefore, this study aims to evaluate the deep learning techniques in recognizing plant diseases using leaf images. Subsequently, about 11 deep learning architectural models were used to identify various types of plant diseases by examining the symptoms that shown in the images.

2. Related works

In 2019, Peng et., al [2] developed a system, which identifies the various diseases that affect apples using a real-time Convolutional Neural Network (CNN) and an architectural model called INAR-SSD (SSD with Inception module and Rainbow concatenation), where previous studies employ images as the essential input. The minimum accuracy value obtained from that study was 70.63%. Xihai et., al [3] improved the deep learning techniques using GoogLeNet and Cifar10 architectures to increase the accuracy and reduce the network parameters. Subsequently, the dataset used was several corn leaf images obtained by independent acquisition from PlantVillage at a minimum accuracy of about 85.4%.

Received: October 8th, 2020. Accepted: October 12nd, 2021
DOI: 10.15676/ijeei.2021.13.4.5

Meanwhile, Uday et., al. [4] used a multilayer convolutional neural network (MCNN) to identify mango leaf diseases at a minimum accuracy value of 88.39%. Rathan et., al [5] also used CNN and the ResNet50 architectural model to identify papaya leaf diseases, with the lowest accuracy value of 83.5%.

Liu et., al [6] proposed improved deep CNN to identification three common kiwifruit leaf diseases. The dataset consisted of 11322 kiwifruit leaf images, which were augmented in order to increase the variation of the data. Additionally, the novel CNN-based model, KiwiConvNet, recorded an accuracy value of about 98.54%, which was better than GoogLeNet and ResNet20. Barman et., al [7] and Iqbal et., al [8] proposed machine learning to detect potato disease. Barman proposes a self-built CNN called SBCNN, which achieved the best accuracy of 99.71%. Meanwhile, Iqbal detected two common potato leaf diseases including Early and Late Blight, using a random forest classifier with 97% accuracy.

Several previous studies have shown that when only one type of plant is used, the lowest accuracy value is always less than 90%. Generally, there are a large number of diseases and plant species in the world. Therefore, this study requires the use of more types of plants and a minimum accuracy level above 90%.

3. Methodology

Several deep learning architectural models are used to identify diseases in leaf images since the different types of plants and diseases are found in the PlantVillage dataset. CNN architecture is the most popular deep learning architecture for image classification. Generally, CNN can be divided into two main parts, namely feature extraction and classification, as shown in Figure 1. Meanwhile, convolution is performed with the inputted image and the number of filters to produce a feature map from raw data.

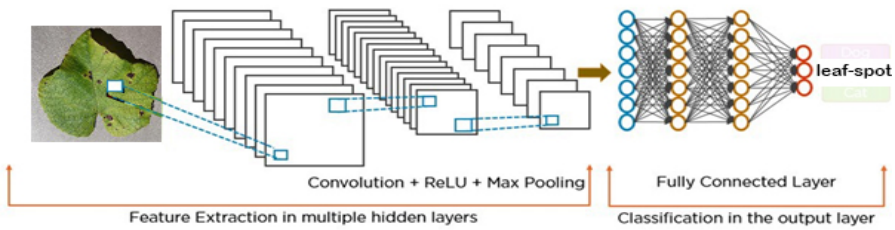

Figure 1. Main parts of CNN

The resulting feature map is subsequently flattened and used by a neural network with a fully connected layer to classify the features obtained during the extraction stage. Also, weight values between the neurons are generated during the training process, while the final stage of the classification involves the use of a softmax classifier, which is used to calculate the probability values for each class from the inputted image.

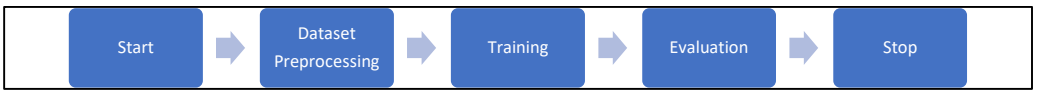

Figure 2. Research block diagram.

Figure 2 shows a block diagram of the conducted study. During the preprocessing stage, the image dimensions were adjusted to the input size for each architectural model used. Furthermore, the pixel intensity value was converted to a floating number (0-1).

Various augmentation techniques were performed, such as rotation, horizontal flip, width and length shifting, shear, and zooming in order to vary the training image.
The 11 deep learning architectural models used in the training stage include:

1. DenseNet121[9]
2. DenseNet169[9]
3. DenseNet201[9]
4. InceptionResNetV2[10]
5. InceptionV3[11]
6. ResNet50V2[12]
7. MobileNetV2[13]
8. NASNetMobile[14]
9. ResNet101V2[12]
10. ResNet152V2[12]
11. Xception[15]

In this study, three scenarios were designed in the evaluation of the architectural model

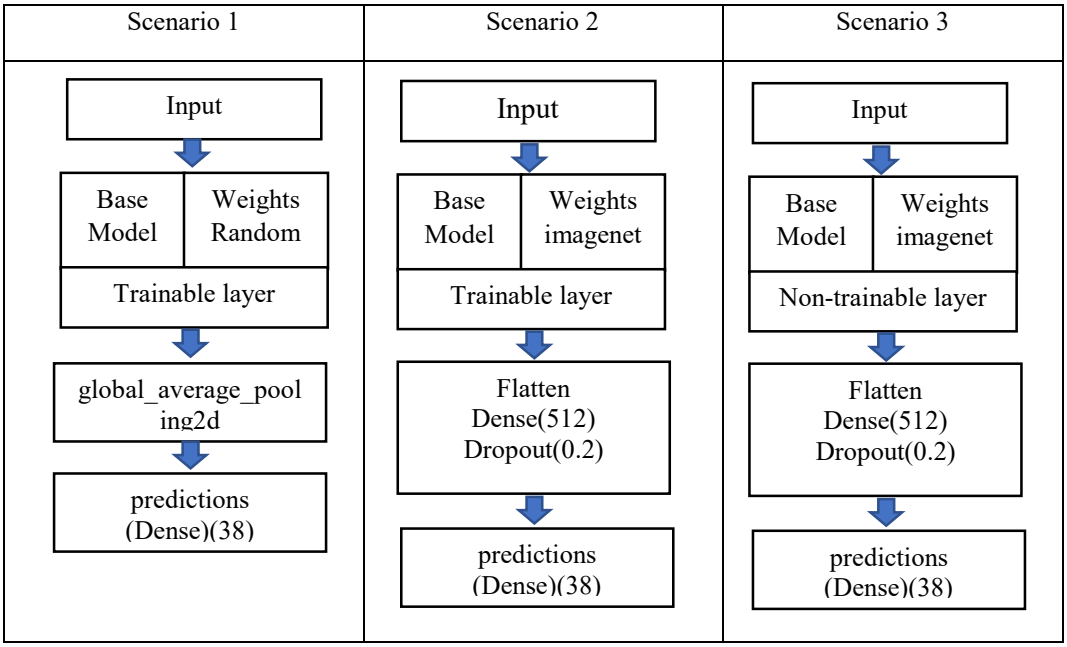

Figure 3. Test scenarios for each architectural model.

Each deep learning architectural model was monitored during the training process using the PlantVillage dataset [16] featuring about 38 plant conditions. Figure 2 shows three test scenarios for each architectural model with the following conditions included:
The training was conducted for each architectural model by introducing initial weights with random values

The training was conducted for each architectural model using the weights from the results of the imagenet dataset, even though the weights of each layer was trainable.
The training was conducted for each architectural model using the weights from the results of the imagenet training dataset, even though the weight of each layer was non-trainable

At the final stage of the evaluation, testing of the trained deep learning architectural model was conducted by involving all images in the training, validation, and test image groups. Furthermore, the percentage accuracy was calculated for each class in the dataset image.

This study uses google colab facilities for the entire study phase, with the hardware specifications as follows:

- GPU Nvidia K80/P100
- GPU Memory 16GB
- GPU clock 1.59 GHz
- CPU core 2
- Ram 12 GB
- Disk Space 358GB

4. Result and Discussion

A. Dataset

This study used the PlantVillage dataset with a total of about 60322 images. Additionally, the dataset was divided into 38 folders representing the various classes of each image. Each folder was named according to its class name, and the images were divided into 3 groups, namely:

- Training data group = 54305 Images
- Validation data group = 5455 Images
- Test data group = 562 images

Each group was divided into 38 classes, which were stored in an array of variables.

```
Class = array(['Apple___Apple_scab', Apple___Black_rot', 'Apple___Cedar_apple_rust',
        'Apple___healthy',
        'Blueberry___healthy','Cherry_(including_sour)___Powdery_mildew',
        'Cherry_(including_sour)___healthy', 'Corn_(maize)___Cercospora_leaf_spot
        Gray_leaf_spot',
        'Corn_(maize)___Common_rust_','Corn_(maize)___Northern_Leaf_Blight','Corn_(
        maize)___healthy','Grape___Black_rot','Grape___Esca_(Black_Measles)','Grape___
        Leaf_blight_(Isariopsis_Leaf_Spot)','Grape___healthy','Orange___Haunglongbing_(
        Citrus_greening)','Peach___Bacterial_spot','Peach___healthy','Pepper,_bell___Bacte
        rial_spot',
        'Pepper,_bell___healthy','Potato___Early_blight','Potato___Late_blight','Potato___h
        ealthy', 'Raspberry___healthy', 'Soybean___healthy','Squash___Powdery_mildew',
        'Strawberry___Leaf_scorch','Strawberry___healthy',
        'Tomato___Bacterial_spot','Tomato___Early_blight',
        'Tomato___Late_blight','Tomato___Leaf_Mold',
        'Tomato___Septoria_leaf_spot','Tomato___Spider_mites Two-
        spotted_spider_mite','Tomato___Target_Spot',
        'Tomato___Tomato_Yellow_Leaf_Curl_Virus','Tomato___Tomato_mosaic_virus',
        'Tomato___healthy'], dtype='<U50')
```

The images in the PlantVillage dataset were captured perpendicular to the camera and the leaf under controlled conditions in the laboratory. Furthermore, the background of image capture was also made plain since there were no other objects in the picture. Figure 3 showed an example of a leaf image in the PlantVillage dataset.

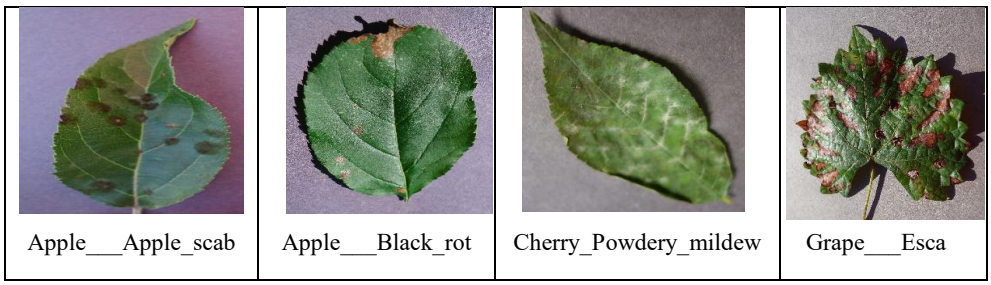

Figure 4. Example of leaf image in the PlantVillage dataset

B. Preprocessing

Before the training process, the dataset image was preprocessed in the following steps:
Step 1. Resize the image dimensions according to the model input size

Step 2. Rescale input from 0-255 to 0-1
Step 3. Augmentation

- Rotation with a random angle between 0-40^{o}
- Flip horizontal
- Width shift
- Height shift
- Shear
- Zoom

C. Training

After the completion of the preprocessing, the image is included in a training stage. Table 1 showed the number of trainable and non-trainable parameters for each model in the three scenarios. For scenario 1, MobileNetV2 model had the smallest number of trainable parameters, and ResNet152V2 has the largest. Meanwhile, in scenarios 2 and 3, the smallest and largest number of trainable parameters were NASNetMobile and ResNet50V2, respectively.

Table 1. The number of parameters for each model in the three training scenarios.

No	Model	Scenario 1		Scenario 2		Scenario 3	
		Trainable params	Non-trainable params	Trainable params	Non-trainable params	Trainable params	Non-trainable params
1	DenseNet121	6,992,806	83,648	49,441,190	83,648	42,487,334	7,037,504
2	DenseNet169	12,547,750	158,400	81,513,894	158,400	69,029,414	12,642,880
3	DenseNet201	18,165,926	229,056	97,739,174	229,056	79,646,246	18,321,984
4	InceptionResNetV2	54,334,598	60,544	104,627,846	60,544	50,351,654	54,336,736
5	InceptionV3	21,846,214	34,432	88,897,222	34,432	67,128,870	21,802,784
6	ResNet50V2	23,597,222	45,440	128,396,966	45,440	104,877,606	23,564,800
7	MobileNetV2	2,272,550	34,112	67,779,878	34,112	65,556,006	2,257,984
8	NASNetMobile	4,273,144	36,738	30,745,912	36,738	26,512,934	4,269,716
9	ResNet101V2	42,606,758	97,664	93,929,126	97,664	51,400,230	42,626,560
10	ResNet152V2	58,265,766	143,744	109,588,134	143,744	51,400,230	58,331,648
11	Xception	20,884,814	54,528	72,207,182	54,528	51,400,230	20,861,480

The number of trainable and non-trainable parameters are associated with the computational load during the training process, where the higher the number of the parameters, the greater the computational load.

Table 2 showed the computation time for one epoch in each model. The table reported that MobileNetV2 model has the shortest computation time in the three training scenarios. Furthermore, one epoch may be completed in about ± 11 minutes, while the longest computation time occurs in the Xception model by spending about ± 25 minutes per epoch.

Table 2. Average time for each model in the three training scenarios.

No	Model	Average training time per epoch (second)		
		Scenario 1	Scenario 2	Scenario 3
1	DenseNet121	680	721	626
2	DenseNet169	803	821	744
3	DenseNet201	895	905	746
4	InceptionResNetV2	790	851	666
5	InceptionV3	1215	1331	1018
6	ResNet50V2	701	713	629
7	MobileNetV2	670	700	614
8	NASNetMobile	754	805	683
9	ResNet101V2	864	814	975
10	ResNet152V2	955	882	1147
11	Xception	1578	1594	1227

The time required for the training process in each epoch affects the maximum number of epochs to be trained using google collab since the runtime is limited to 12 hours. The maximum number of epochs is 28 when using the longest time value of 25 minutes/epoch. The training process for all scenarios in this study is limited to the first 20 epochs. However, previous research has shown that the training process has reached convergence in less than 20 epochs.

D. Evaluation

At the evaluation stage, the training model was tested by involving all image groups, which included the training, validation, and test groups. In addition, the number of true and false predicted values for each class in each scenario was tabulated. Table 1 showed the data for the accuracy level of each mode in Scenario 1. The results showed that almost all models have a maximum accuracy percentage close to 100%, but the minimum accuracy ranged from 0.87 % to 87.1 %, which was 0.87 % for the NASNetMobile model was 0.87 %. Meanwhile, the InceptionResNetV2 model had the highest minimum accuracy and an accuracy percentage <90% in only one class.

The accuracy level of the result of the training model in scenario 2 is shown in table 4. The minimum accuracy values in the models were within the range of 0%-77.4%, with the lowest values seen in the ResNet152V2 model. Similar to scenario 1, the InceptionResNetV2 model had the highest minimum accuracy when compared to others. However, where there was a decrease in the accuracy value to 77.4% in scenario 2, the number of classes with an accuracy level less than 90% was increased by 4.

Table 3. The accuracy of the training result model in scenario 1.

No	Model	Accuracy (percent)			Number of classes with an accuracy of <90%
		Min	Max	Average	
1	DenseNet121	82.18	100	98.65	2
2	DenseNet169	68.24	100	96.87	2
3	DenseNet201	65.45	100	93.80	9
4	InceptionResNetV2	87.10	100	98.86	1
5	InceptionV3	29.05	100	93.69	6
6	ResNet50V2	27.44	100	86.45	12
7	MobileNetV2	48.94	100	93.66	8
8	NASNetMobile	0.87	99.76	68.51	25
9	ResNet101V2	37.65	100	91.78	11
10	ResNet152V2	10.62	100	75.81	18
11	Xception	4.32	100	67.16	28

Table 4. The accuracy of the training result model in scenario 2.

No	Model	Accuracy (percent)			Number of classes with an accuracy of <90%
		Min	Max	Average	
1	DenseNet121	35.9	99.8	84.6	18
2	DenseNet169	11.03	99.4	75.85	27
3	DenseNet201	4.12	98.8	63.3	34
4	InceptionResNetV2	77.40	100	97.25	4
5	InceptionV3	70	99.93	94.44	7
6	ResNet50V2	9.41	100	81.5	14
7	MobileNetV2	13.46	100	86.94	15
8	NASNetMobile	0.99	98.75	59.14	27
9	ResNet101V2	42.36	100	90.76	11
10	ResNet152V2	0	91.95	20.70	37
11	Xception	73.56	100	94.25	8

Table 5 showed the results of the accuracy for the training model in scenario 3. In this scenario, the pre-train model uses ImageNet training weights. However, all the layers in the set are non-trainable, and the ImageNet weight values are kept unchanged during the training process. Generally, the minimum accuracy value in the third scenario was lower compared to the two previous scenarios. Also, the minimum value range was 0%-42.94%, with the lowest values seen in the InceptionV3 and InceptionResNetV2 models at 0%. However, ResNet101V2 obtained the highest minimum value, unlike the two previous scenarios.

Table 5. The accuracy of the training result model in scenario 3.

No	Model	Accuracy (percent)			Number of classes with an accuracy of <90%
		min	max	average	
1	DenseNet121	27.32	99.90	88.70	13
2	DenseNet169	29.31	99.92	92.06	7
3	DenseNet201	38.82	99.88	92.32	9
4	InceptionResNetV2	0	100	76.86	25
5	InceptionV3	0	99.76	67.01	23
6	ResNet50V2	33.71	99.95	89.36	10
7	MobileNetV2	42.73	99.94	89.31	12
8	NASNetMobile	13.21	99.62	81.65	22
9	ResNet101V2	42.94	99.80	90.05	15
10	ResNet152V2	39.28	100	87.98	15
11	Xception	24.91	99.92	86.33	15

The test results proposed in this study showed that the highest minimum value generally occurs in scenario 1, where the weight of the pre-train model was initiated by a random value. Furthermore, although the InceptionResNetV2 model achieved a minimum accuracy value of 87.10%, only 1 of the 38 classes had an accuracy level less than 90%.

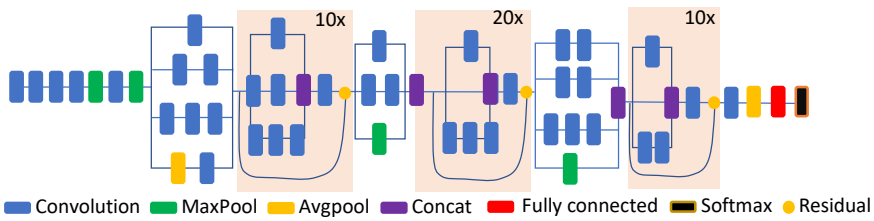

Figure 5. Schematic diagram of Inception-ResNet-v2

The schematic of the InceptionResNetV2 architectural model was shown in Figure 4. This model has a deeper network when compared to other architectural models. Additionally, the total number of trainable and non-trainable parameters was 54,395,142, which was higher in some other models.

In addition to the accuracy level, the F1-score can also be used to properly evaluate the performance of the architecture models.

Table 6. The F1-score in three scenarios

No	Model	F1-score		
		Scenario 1	Scenario 2	Scenario 3
1	DenseNet121	0.98716	0.97968	0.96946
2	DenseNet169	0.95429	0.72659	0.92352
3	DenseNet201	0.94578	0.67667	0.91777
4	InceptionResNetV2	0.97991	0.95433	0.77513
5	InceptionV3	0.99371	0.99126	0.93592
6	ResNet50V2	0.96363	0.75910	0.90242
7	MobileNetV2	0.48283	0.83193	0.89455
8	NASNetMobile	0.02112	0.59884	0.82948
9	ResNet101V2	0.02622	0.92535	0.90755
10	ResNet152V2	0.89357	0.18920	0.89145
11	Xception	0.70577	0.99419	0.85834

The best F1-score value of 0.99419 occurs in the second scenario of the Xception model. Meanwhile, the worst value of 0.02712 occurred in the first scenario of the NASNetMobile model. The InceptionResNetV2 had the best accuracy value and the highest F1-score of 0.97991 in the first scenario, which is still higher than the F1-score of 0.70577 in the first scenario of the Xception model.

5. Conclusion

In this study, about 11 deep learning architectural models were used to identify diseases in plant leaf images. The test results were compared to determine the model with the highest minimum accuracy value. The results showed that scenario 1 was generally better than scenarios 2 and 3. Also, the InceptionResNetV2 architectural model had the highest accuracy, but one class had an accuracy of less than 90%. However, when using the F1-score approach, the Xception model in the second scenario achieved the highest score. The highest F1-score seen in the InceptionResNetV2 model occurred in the first scenario, which was slightly lower compared to the Xception model. Therefore, the InceptionResNetV2 model architecture is predicted to be the best model for the PlantVillage dataset since it has the highest accuracy and F1 scores. For further study, this model should be conducted with several optimizations since all classes have an accuracy value of more than 90%.

6. Acknowledgment
The authors gratefully acknowledge the support of Universitas Gadjah Mada.

7. References

[1]. S. Sankaran, A. Mishra, R. Ehsani, and C. Davis, "A review of advanced techniques for detecting plant diseases," *Comput. Electron. Agric.*, vol. 72, no. 1, pp. 1–13, 2010.
[2]. P. Jiang, Y. Chen, B. Liu, D. He, and C. Liang, "Real-Time Detection of Apple Leaf Diseases Using Deep Learning Approach Based on Improved Convolutional Neural Networks," *IEEE Access*, vol. 7, pp. 59069–59080, 2019.
[3]. X. Zhang, Y. Qiao, F. Meng, C. Fan, and M. Zhang, "Identification of maize leaf diseases using improved deep convolutional neural networks," *IEEE Access*, vol. 6, pp. 30370–30377, 2018.
[4]. U. P. Singh, S. S. Chouhan, S. Jain, and S. Jain, "Multilayer Convolution Neural Network for the Classification of Mango Leaves Infected by Anthracnose Disease," *IEEE Access*, vol. 7, pp. 43721–43729, 2019.
[5]. R. K. V. B and M. S. Nagugari, *for Classification and Detection*, vol. 1. Springer International Publishing.
[6]. B. Liu, Z. Ding, L. Tian, D. He, S. Li, and H. Wang, "Kiwifruit Leaf Disease Identification Using Improved Deep Convolutional Neural Networks," *Front. Plant Sci.*, vol. 11, pp. 1267–1272, 2020.
[7]. U. Barman, D. Sahu, G. G. Barman, and J. Das, "Comparative Assessment of Deep Learning to Detect the Leaf Diseases of Potato based on Data Augmentation," pp. 682–687, 2020.
[8]. M. A. Iqbal and K. H. Talukder, "Detection of Potato Disease Using Image Segmentation and Machine Learning," pp. 43–47, 2020.
[9]. G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," Aug. 2016.
[10]. C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," Feb. 2016.
[11]. C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," Dec. 2015.
[12]. K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," Dec. 2015.
[13]. M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," Jan. 2018.
[14]. B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning Transferable Architectures for Scalable Image Recognition," Jul. 2017.
[15]. F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," Oct. 2016.
[16]. D. P. Hughes and M. Salathe, "An open access repository of images on plant health to enable the development of mobile disease diagnostics," Nov. 2015.

Harry Yuliansyah received the B.S. degree in Electrical Engineering from Bengkulu University, in 2009. Master degree of Engineering in 2011 from Universitas Gadjah Mada, Indonesia. He joined Institut Teknologi Sumatera (ITERA) as teaching staff in 2012. He is currently pursuing the Doctoral degree with Electrical Engineering Universitas Gadjah Mada, Indonesia. His research interests include image recognition, deep learning, and neural network compression.

Rudy Hartanto received the Bachelor degree of Electrical Engineering in 1989, Master degree of Engineering in 1995, and Doctor in 2015 from Universitas Gadjah Mada, Indonesia. He started to join as teaching staff in the Department of Electrical Engineering and Information Technology, Faculty of Engineering, Universitas Gadjah Mada (UGM) in 1990. Currently, he is an Associate Professor and Head of Study Program of Magister Information Technology, Faculty of Engineering, Universitas Gadjah Mada (UGM). His current research interests include computer graphics, computer vision, computer-human interaction, image processing, multimedia programming, signal processing, virtual reality and related simulation

Indah Soesanti received both M.Eng. and Ph.D. degrees from Department of Electrical Engineering, Universitas Gadjah Mada, Indonesia in 2001 and 2011, respectively. She is a Lecturer in the Department of Electrical Engineering and Information Technology, Faculty of Engineering, Universitas Gadjah Mada, Indonesia. Her current research interests are signal and image processing, biomedical image analysis, optimization, artificial intelligence, decision support system, and pattern classification.