Diseases Classification for Tea Plant Using Concatenated Convolution Neural Network

Dikdik Krisnandi[1], Hilman F. Pardede[2], R. Sandra Yuwana[3], Vicky Zilvan[4], Ana Heryana[5], Fani Fauziah[6], and Vitria Puspitasari Rahadi[7]
[1−5]Research Center for Informatics (P2I) - Indonesia Institute of Sciences (LIPI)
Bandung 40135, Indonesia
[6−7]Research Institute for Tea and Cinchona - Indonesian Agency for Agricultural Research and Development
Gambung 40972, Indonesia
Email: [1]dikd003@lipi.go.id, [2]hilm001@lipi.go.id, [3]rade014@lipi.go.id, [4]vick001@lipi.go.id, [5]anah002@lipi.go.id, [6]fani.fauziah@ritc.id, [7]vitria.rahadi@ritc.id

***Abstract*—Plant diseases can cause a significant decrease in tea crop production. Early disease detection can help to minimize the loss. For tea plants, experts can identify the diseases by visual inspection on the leaves. However, providing experts to deal with disease identification may be very costly. The machine learning technology can be implemented to provide automatic plant disease detection. Currently, deep learning is state-of-the-art for object identification in computer vision. In this study, the researchers propose the Convolutional Neural Network (CNN) for tea disease detections. The researchers focus on the implementation of concatenated CNN, namely GoogleNet, Xception, and Inception-ResNet-v2, for this task. About 4727 images of tea leaves are collected, comprising of three types of diseases that commonly occur in Indonesia and a healthy class. The experimental results confirm the effectiveness of concatenated CNN for tea disease detections. The accuracy of 89.64% is achieved.**

***Index Terms*—Concatenated Convolution Neural Network, Classification, GoogLeNet, Xception, Inception-ResNet-v2**

I. Introduction

TEA (*Camellia sinensis*) is one of the major agricultural commodities in Indonesia. However, some of the tea clones are susceptible to pests and diseases. In Indonesia, the main disease that frequently attacks tea plants is *Exobasidium vexans Massee* causing a disease called blister blight, leafhoppers (*Empoasca sp.*), and the looper caterpillar (*Hyposidra talaca*). The pests and diseases have different characteristics that can be distinguishable from the leaves. However, experts are still required to manually identify them since some pest and disease symptoms may look similar. Nevertheless, providing enough experts to deal with vast areas of plantations is very costly and impossible.

Received: Aug. 12, 2019; received in revised form: Sept. 30, 2019; accepted: Oct. 08, 2019; available online: Oct. 29, 2019.

The machine learning technology is useful to develop a device for the automatic detection of plant diseases. The application can help in early disease detection to minimize the risk of crop failure. It can also be used as inputs for sorting the harvest production to identify the quality of tea. Plant disease detection can be categorized as classification tasks in machine learning. Classification is a grouping of data for each target class. Classification algorithms are usually trained in a supervised manner. For supervised learning, the relations between the features of the data and their class labels are assumed to follow the presumed classification algorithms. Then, during training, the optimum sets of hyper-parameters that minimize the loss function, such as the mean squared error of the model, are selected. Thus, the challenges in classifiers with good performances are to find the best features or classification models.

For object recognition tasks, there have been various studies that propose good features for object detection or classification. The traditional machine learning techniques are usually used as input to the algorithm. The examples of these techniques are Scale Invariant Feature Transform (SIFT), Speed up Robust Feature (SURF), and Histograms of Oriented Gradients (HOGs). SIFT is very efficient in object recognition applications, but it requires a large computational complexity [1]. SURF technique has performed faster than SIFT and detected points without reducing. HOGs are feature descriptors for object detection [2]. HOG counting calculates gradients in parts of an image [3]. These three examples of traditional techniques require complex calculations and are more challenging to apply to online applications. However, most of these features are computationally complex.

For classifiers, Support Vector Machine (SVM) is arguably one of the most popular classifiers for object recognition before the era of deep learning. Reference [4] proposed SVM for the recognition and detection of tea leaf diseases. Then, Ref. [5] used SVM combining K-Nearest Neighbor (KNN) with geometric moment invariant to increase the results of image recognition. Moreover, combining the SVM with Kernel Principal Component Analysis (KPCA) reduces the dimensional feature vector [6]. However, SVM is also usually tweaking using kernel functions to find good performance, and this is not always easy to find.

Currently, deep learning is a popular technique for many tasks for object recognition. Reference [7] reviewed object detection frameworks that use deep learning. They focused on typical generic object detection architectures with modifications to improve performance. Then, Ref. [8] used CaffeNet architecture to recognize plant disease. They altered the last layer and the output of the softmax layer to support 15 classes. Deep learning models nonlinear relations between data and their class labels used a stacked multi-layer perceptron. Hence, it theoretically could fit any functions if there were enough depth and number of neurons. Therefore, even when it was given simple and raw features, the studies found that deep learning could achieve good performance.

Machine learning methods have been used in some studies for plant disease detection. Disease classification of plants was carried out by Ref. [9] on the images of cucumber leaves. They used K-means clustering and Sparse Representation (SR). AlexNet and VGG16 net are used by Ref. [10] to classify images of tomato leaves in six disease classes and healthy classes with a dataset from PlantVillage. Reference [11] used a deep Convolution Neural Network (CNN) on a healthy leaf dataset of 54306 images from PlantVillage to identify 14 crop species and 26 diseases. Identification of symptoms of disease in cassava leaves is carried out by Ref. [12] using CNN model.

Reference [13] identified apple leaf disease using AlexNet. The dataset used was 13.689 images of apple leaves and achieved an accuracy of 97.62%. Reference [14] used Deep CNN (DCNN) to identify leaf diseases in 39 classes. The accuracy reached 96.46%. Meanwhile, Ref. [15] used 87.848 images to detect and diagnose plant diseases using the CNN model. Its success reached 99.53%. Reference [16] diagnosed leaf disease automatically with an accuracy of 93.85%, using two feature extraction techniques, namely Gray Level Covariance Matrix (GLCM) and Alexnet.

Recently, deep learning methods are becoming more popular as the methods for plant disease detection. Reference [11] identified 14 plant species and 26 types of diseases in plants, using the various CNN architectures. Reference [17] tried to classify the introduction of disease in plants by using deep learning on 50.000 images of corn. Moreover, Ref. [18] used a CNN and Deconvolutional Network (DN) for identifying plant species. Reference [19] tried to classify 22 species of weeds and plants using DCNN.

According to Ref. [20], diseases in plants can be caused by bacteria, viruses, and fungi. Viral diseases are the most difficult to diagnose and control their spread. The characteristics of the plants affected by the virus can be observed from the leaves. It becomes tangled and curly and has stunted growth. Small pale spots usually characterize leaves that are attacked by bacteria. For fungi, it will be easily identified through its morphological characteristics.

Tea disease identification has been proposed in some studies. Reference [21] used 26 tea plant samples with typical discoloration symptoms from different tea gardens. They conducted a metagenomic analysis based on next-generation sequencing. Reference [22] tried to identify diseases in tea plants. The disease included algal leaf spots, brown blight, gray blight, blister blight, horsehair blight, twig dieback, and canker stem.

Similarly, Ref. [23] used CNNs to recognize plant diseases using images of tea leaves automatically. A CNN model called LeafNet was proposed in the study. It was a sequential model in which CNN was stacked on the top of the preceding layers, and the flow of information only went in one direction.

Reference [24] conducted a study for the introduction of disease in tea leaves by using the Neural Network Ensemble (NNE) for pattern recognition. The study also implemented sequential networks with a multi-layer perceptron. However, the sequential model might have limitations, especially when networks with many layers were implemented. First, the networks might lose some information due to the implementation of pooling layers. This issue might apply to the case of CNN. Second, networks might need more data for training due to the high number of parameters to be optimized. Lastly, the parameters of the networks were more susceptible to a local minimum.

For these reasons, the researchers choose concatenated CNN for this task. The concatenated network allows the flow of information to have more than one direction. By doing so, the researchers can allow one flow of information to carry information from previous networks that may have been lost. In addition, previous researchers find that using a concatenated network makes the parameters less susceptible to a local minimum. Previous researchers have strongly

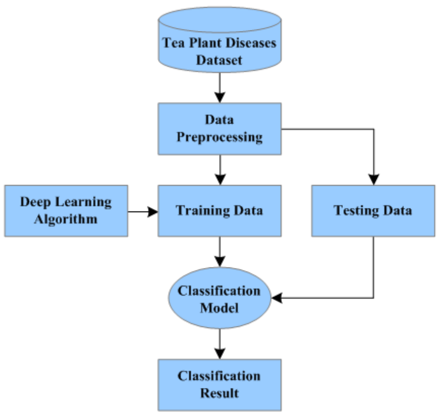


Fig. 1. The flow diagram of the disease identification system in the tea plant.

indicated that the concatenated CNN achieves better performance than the sequential model [25–29]. In addition, to concatenate, deep residual learning is also used to improve performance [28, 29]. The researchers apply three concatenated CNN architectures in this study, namely GoogLeNet, Xception, and Inception-ResNet-v2.

II. Research Method

The flow diagram of this research method can be seen in Fig. 1. The researchers divided the group of data into four data class labels. The first data classes are combined between healthy tea plants and blister blight. The second data classes consist of the healthy tea plant, blister blight, and Empoasca sp. Then, the last data classes are the combination of all data with healthy tea plants, blister blight, *Empoasca sp.*, and looper caterpillar.

The researchers choose 80% of the data for training data, 10% for data validation, and 10% for testing data. Three architectures are used in this study for concatenated CNN. Those are GoogLeNet, Xception, and Inception-ResNet-v2. All architectures use the same training and validation sets with the Rectified Linear Units (ReLU) activation function. The batch size is 10.

A. Convolution Neural Network

CNN is a commonly used architecture for many tasks in object recognition. It is a variant of Multi-Layer Perceptron (MLP). The nodes of succeeding layers in MLP are all connected to all the nodes from preceding layers, but it is not the case on CNN. The nodes are only connected to some neighboring nodes of preceding layers. This benefits the object recognition tasks, which may learn nearby pixels of the image to determine the class objects. For another advantage of CNN for image data, it is designed to process two or more-dimensional data so that each neuron on CNN is presented in two or more-dimensional form. Data that propagates on CNN has a linear operation and different weighting parameters. Linear operations on CNN use convolution with four-dimensional weights, which are the convolution kernel assemblies. In this CNN algorithm, the input from the previous layer is not a one-dimensional array but a two-dimensional array. The excess of CNN that uses dimensions more than one affects the overall scale in an object. The entire scale of the object is significant so that the input does not lose its spatial information, which will extract and classify the features. Thus, the CNN algorithm will increase the level of accuracy and optimization.

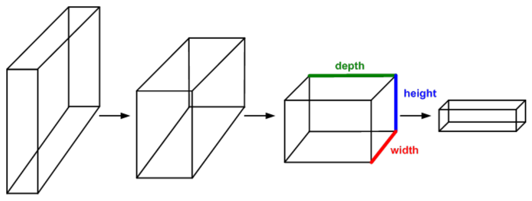


Fig. 2. The three dimensions of the Convolutional Neural Network.

CNN forms its neurons in three dimensions (length, width, and height) in a layer. Figure 2 shows CNN in three dimensions in one of the layers.

B. Concatenated Convolution Neural Network

Many studies that implemented CNN focus on depth. The examples of CNN using DCNN include LeNet, AlexNet, ZFNet, VGGNet, and ResNet. However, very deep networks are prone to overfit. Meanwhile, stacking large convolution operations is computationally expensive. For this reason, a concatenated CNN module is developed. The module does not only focus on the depth factor but also the width factor. The examples of concatenated CNN are GoogLeNet and Xception.

The concatenated CNN concept is developed from the inception module. The inception module is used for more efficient computational and deeper networks. It reduces a dimension with stacked 1×1 convolution. Figure 3 shows the concept of the inception module. Convolution in the input is executed with three different filter sizes: 1×1, 3×3, and 5×5. Additionally, max pooling is also executed. The outputs are concatenated and sent to the next inception module.

One of the problems with training networks, especially deep neural networks, is that it vanishes and explodes the gradients. The problem of vanishing or

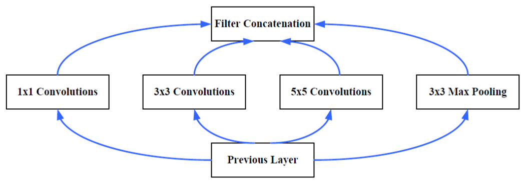


Fig. 3. The inception module.

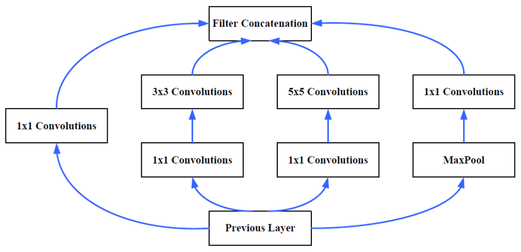


Fig. 4. The inception module with dimension reduction in GoogLeNet.

exploding gradients for a long time is a big barrier to train deep neural networks. In training a very deep network, it obtains very large, very small, or even exponentially small derivatives or slopes. Thus, this makes training difficult. The concatenated concept can reduce these problems.

In this research, the researchers use three kinds of concatenated CNN architecture. Those are GoogLeNet, Xception, and Inception-ResNet-v2.

C. GoogLeNet

The GoogLeNet architecture is proposed by Ref. [26]. GoogLeNet has optimized depth and the width factor of the network to raise accuracy. To improve the performance of deep neural networks, it is usually by increasing the size of the network. However, if the size of the network is increased, it will increase the number of parameters to be trained. GoogLeNet solves this problem by using the inception module. The inception module deploys a parallel combination of convolution. Then, 1×1 convolutions are used for computing the reduction before 3×3 and 5×5 convolutions. GoogLeNet has nine inception modules stacked linearly. It consists of 22 deep layers and has a lower 5 million parameters in the networks. The schematic module of inception with dimension reduction in GoogLeNet is depicted in Fig. 4.

D. Xception

An Xception, which stands for extreme inception, is first proposed by Ref. [27]. Xception is a CNN

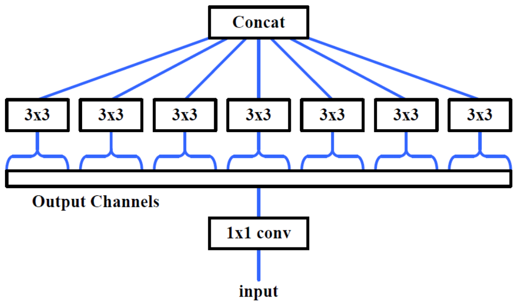


Fig. 5. The extreme form of inception module.

architecture based entirely on depthwise separable convolution layers. It has 36 convolutional layers. The 36 convolutional layers are constructed into 14 modules. Except for the first and last modules, all modules have linear residual connections.

Exception architecture is based entirely on convolutional layers that can be separated in depthwise. Convolutionally separated module is almost identical to the extreme form of the initial module as depicted in Fig. 5. The difference between convolution that can be separated in depthwise and the extreme beginning is convolution, which can be separated from doing wise spatial convolution first and 1×1 convolution. Meanwhile, the inception performs 1×1 convolution first. Then, Xception architecture can be seen in Fig. 6.

E. Inception-ResNet-v2

The Inception-ResNet-v2 is first introduced by Ref. [29]. Inception-ResNet-v2 architecture combines the principles of the structure of Inception and residual connections. Convolution filters are combined with residual connections evading degradation problems caused by deep structures. Residual connections can reduce training time. Thus, it is more efficient than without using residual connections. Figure 7 presents the schema and stem for the Inception-ResNet-v2 network.

Figure 8 shows the general scheme for scaling combined Inception-ResNet module. For arbitrary subnetworks, the inception block will be replaced using a scaling block that only scales the last linear activation with the appropriate constant. Reducing residue before adding to previous layer activation can stabilize training. In general, some residual scaling factor values are used between 0.1 and 0.3 before being added to the activation of the accumulated layers.

III. Experimental Setup

In this study, the researchers collect 4727 images of tea leaves. The data are collected from plantations at

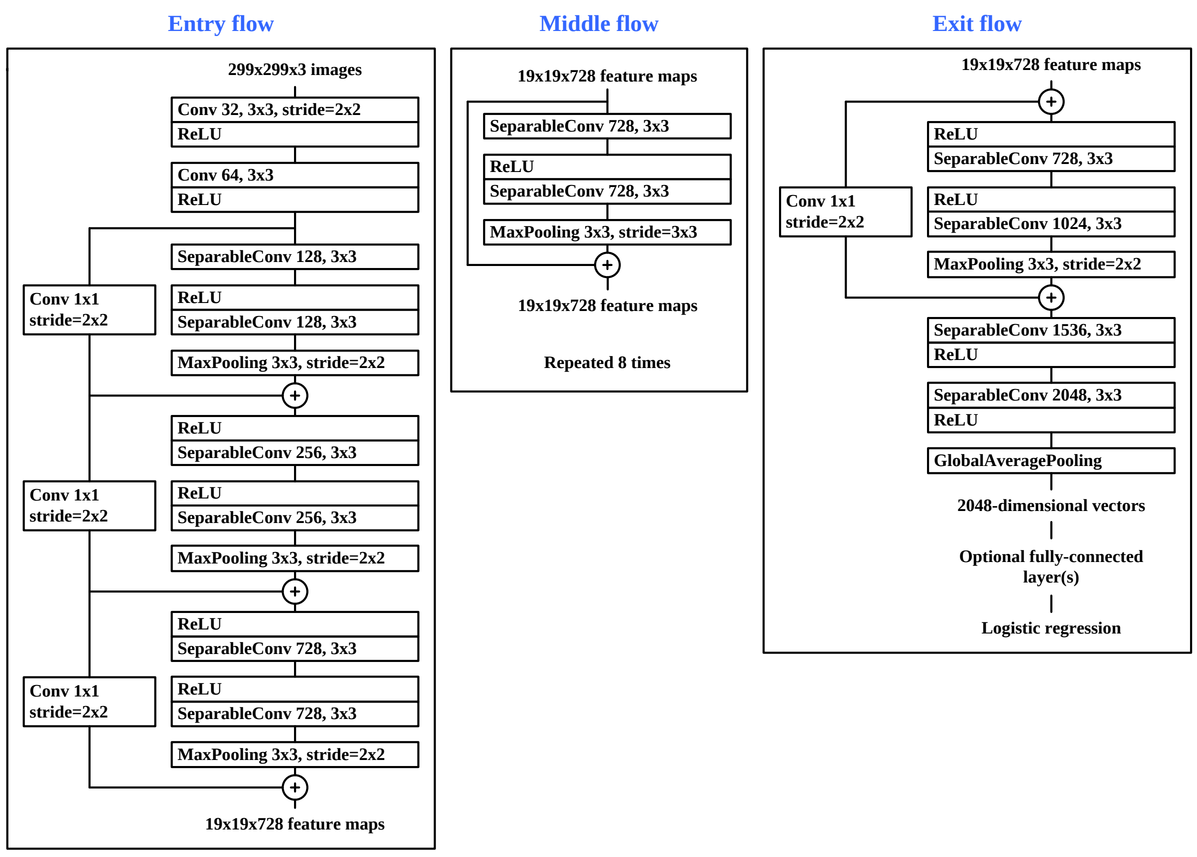


Fig. 6. Xception architecture.

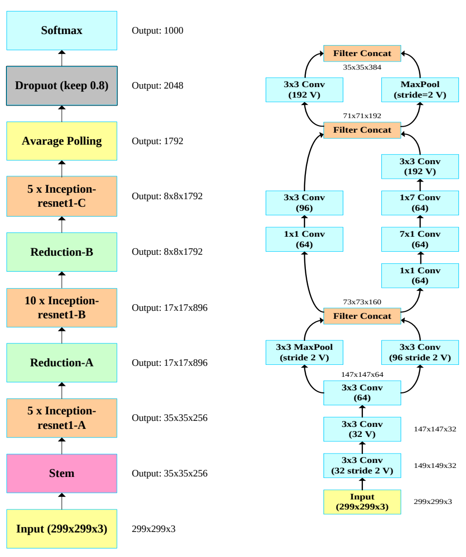


Fig. 7. Schema (left) and stem (right) of the Inception-ResNet-v2 network.

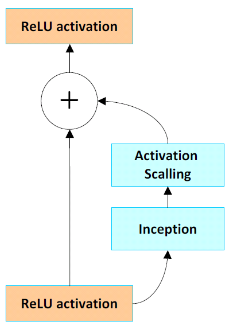


Fig. 8. The general schema of scaling combined Inception-ResNet modules.

The Research Institute for Tea and Cinchona, Gambung, West Java, Indonesia. The collected tea leaves belong to the genus Camellia sinensis and Assamnica species. This dataset is collected using two digital cameras and five smartphone cameras. All images are taken indoor and under an uncontrolled environment. The distance of the camera to the leaves is not determined and uses the autofocus feature.

The researchers collect the data with four class targets comprising of three types of diseases and a healthy class. The dataset consists of 3479 diseased

TABLE I
THE DATASET USED FOR THE CLASSIFICATION

No	Type of Tea Plant Diseases	Number	%
1	Healthy	1248	26.40
2	Blister blight	842	17.81
3	*Empoasca sp.* (leafhoppers)	1728	36.56
4	Looper caterpillars	909	19.23
	Total	4727	100

TABLE II
THE TABULAR REPRESENTATION DATASET USED FOR CLASSIFICATION.

No	Type of Tea Plant Diseases	Number of Training Images	Number of Testing Images
1	Healthy leaf	998	250
2	Blister blight	674	168
3	*Empoasca sp.* (leafhoppers)	1382	346
4	Looper caterpillars	727	182
	Total	3781	946

TABLE III
THE RECOGNITION PERFORMANCE ACCURACY WITH RMSPROP OPTIMIZER AND LEARNING RATE = 10^{-5}.

Data Classes	Architecture	Accuracy (%)		
		Epoch		
		50	100	200
2	GoogLeNet	91.39	94.02	95.22
	Xception	92.82	95.22	94.74
	Inception-ResNet-v2	93.30	96.17	95.69
3	GoogLeNet	74.43	77.36	81.41
	Xception	83.77	82.72	85.86
	Inception-ResNet-v2	84.82	80.11	84.56
4	GoogLeNet	70.19	74.42	78.65
	Xception	80.76	82.42	84.36
	Inception-ResNet-v2	75.69	82.24	83.30

tea plants and 1248 healthy tea plants. For leaves with diseases, the researchers collect leaves that from tea plants with blister blight, *Empoasca sp.* (leafhoppers), and looper caterpillars. For each disease, there are 842 data for blister blight, 1728 data for *Empoasca sp.*, and 909 data for plants with looper caterpillar. The details of data distributions are shown in Table I. Meanwhile, the samples of the collected data are portrayed in Fig. 9.

For the experiment, the researchers resize all the images into fixed 64 × 64 pixels and extract values of RGB from images as features. The data are divided into three subsets: training, validation, and testing sets. The researchers select 80% of the data for training, 10% for validating, and 10% for testing. A tabular representation of the training images in this research is shown in Table II.

The researchers have a total of 108 experimental configurations, which vary on the following parameters:

1) Architecture: GoogLeNet, Xception, and Inception-ResNet-v2.
2) Activation function: ReLU. It is one of the activation functions used in deep learning. The function returns to 0 if it receives any negative input. However, for any positive x value, it returns that value back. It is a simple function, but it can allow a model to account for non-linearities and interactions well. This is the reason why the researchers choose ReLU as an activation function.
3) Optimizer: RMSprop and Adam. RMSprop deals with its radically diminishing learning rates. It keeps the moving average of the squared gradients for each weight and divides the gradient by square root of the mean square. This is why it is called RMSprop (root mean square). Moreover, Adam is an adaptive learning rate optimization algorithm that is designed specifically for training deep neural networks [30]. Adam is a method of the adaptive learning rate. It calculates the rate of individual learning for different parameters. The name comes from the estimation of adaptive moments and uses the first and second-moment gradient estimates to adapt the learning rate for each weight of the neural network.
4) Batch size: 10
5) Epoch: 50, 100, and 200
6) Learning rate: 10^{-5} and 10^{-4}

Then, the researchers implement the concatenated CNN training in python using Tensorflow and Keras package.

IV. RESULTS AND DISCUSSION

The results of the experiments are shown in Tables III–VI. The results are the accuracy of the test data. The results clearly show that the performance of Xception and Inception-ResNet-v2 is better than GoogLeNet. This is expected since Xception and Inception-ResNet-v2 are CNN with significantly more complex performance and have more layers. However, the results show that the performance can be further improved by adding more training data. It is indicated that performance tends to decrease when the researchers use more class labels. The best result for two data class diseases achieves 98.09% of accuracy using a 10^{-4} learning rate. Meanwhile, the results using three data classes are 90.05% of accuracy using the 10^{-4} learning rate. For the four data classes, the best correctly classified data obtain 89.64% of accuracy using the 10^{-4} learning rate.

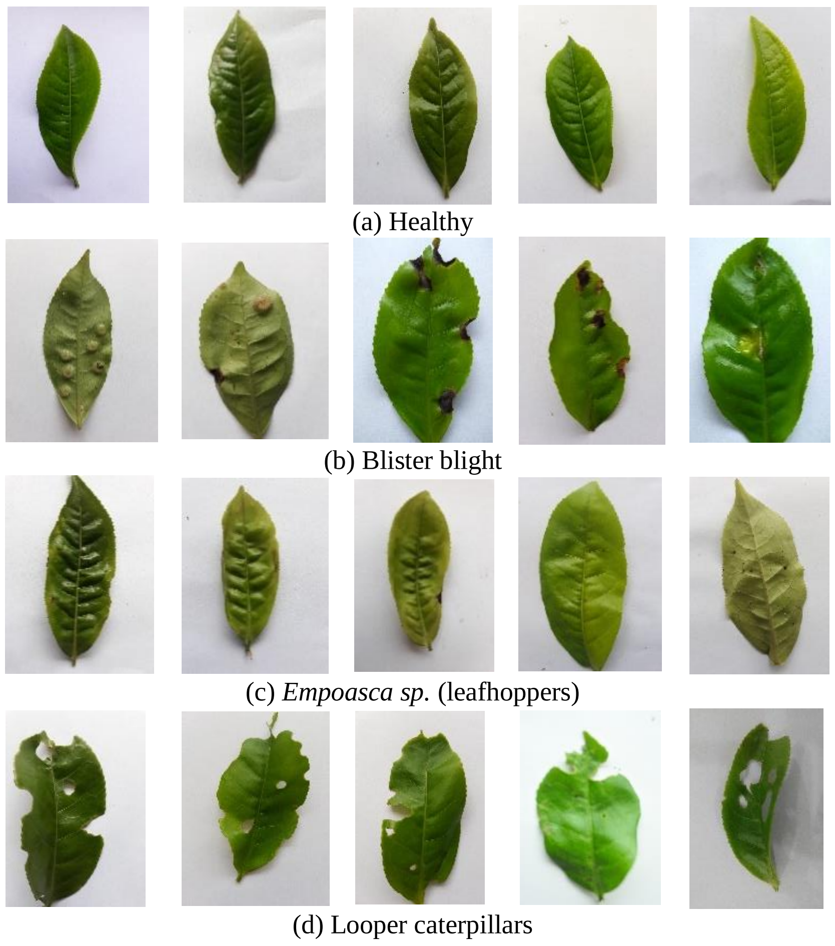


Fig. 9. The examples of the tea plant from the dataset: (a) Healthy, (b) Blister blight, (c) *Empoasca sp.* (leafhoppers), and (d) Looper caterpillars.

TABLE IV
THE RECOGNITION PERFORMANCE ACCURACY WITH ADAM OPTIMIZER AND LEARNING RATE = 10^{-5}.

Data Classes	Architecture	Accuracy (%)		
		Epoch		
		50	100	200
2	GoogLeNet	93.06	94.74	95.69
	Xception	93.78	96.65	92.17
	Inception-ResNet-v2	91.39	95.69	96.65
3	GoogLeNet	75.92	78.80	80.89
	Xception	83.77	84.29	85.86
	Inception-ResNet-v2	76.96	79.32	79.84
4	GoogLeNet	76.74	74.42	77.06
	Xception	82.66	84.99	84.66
	Inception-ResNet-v2	77.38	80.55	80.76

TABLE V
THE RECOGNITION PERFORMANCE ACCURACY WITH RMSPROP OPTIMIZER AND LEARNING RATE = 10^{-4}.

Data Classes	Architecture	Accuracy (%)		
		Epoch		
		50	100	200
2	GoogLeNet	94.26	94.26	93.30
	Xception	96.17	95.69	97.61
	Inception-ResNet-v2	91.87	97.61	97.13
3	GoogLeNet	80.37	84.16	80.89
	Xception	84.82	82.46	82.20
	Inception-ResNet-v2	82.20	89.01	90.05
4	GoogLeNet	74.84	78.65	78.22
	Xception	84.57	83.93	87.10
	Inception-ResNet-v2	84.36	88.80	87.95

In general, the results show that most of the use of Adam optimizer has outperformed the RMSProp optimizer. Adam optimizer is considered to be able to handle sparse gradients on noisy problems. Bias-correction happens towards the end of the optimization as gradients become sparser. Adam optimizer also computes individual learning rates for different parameters.

TABLE VI
THE RECOGNITION PERFORMANCE ACCURACY WITH ADAM OPTIMIZER AND LEARNING RATE = 10^{-4}.

Data Classes	Architecture	Accuracy (%)		
		Epoch		
		50	100	200
2	GoogLeNet	91.63	94.26	94.50
	Xception	96.17	95.22	95.22
	Inception-ResNet-v2	81.82	98.09	97.13
3	GoogLeNet	83.12	84.16	82.20
	Xception	84.03	84.82	84.55
	Inception-ResNet-v2	85.34	76.70	89.27
4	GoogLeNet	77.59	77.91	78.33
	Xception	81.82	82.88	81.61
	Inception-ResNet-v2	85.41	89.64	86.89

The number of epochs used to turn out not to be too significant for accuracy. The more epochs of data training increase accuracy. However, it does not apply linearly. This result is expected since the network may be overfitting when the researchers use a large number of epochs. It may be interesting to see the effect of using regularization in a future study to mitigate the problems.

The progression of the accuracy of the three models is shown in Fig. 10. Figure 10a shows the validation accuracy increasing rapidly in the first five epochs. After that, it is slowly ascended. Validation accuracy is relatively fluctuating until it finally tends to be stable around 189 to 200 epochs. For the GoogLeNet model, it converges slower than Xception and Inception-ResNet-v2. Based on the figure and number, GoogLeNet requires a larger number of epoch than Xception and Inception-ResNet-v2.

Meanwhile, Xception and Inception-ResNet-v2 converge much faster than GoogleNet (Fig. 10a and b). For both architectures, the accuracy of training data rapidly increases around the twenty-fifth epoch. It remains relatively the same afterward.

For Xception, Fig. 10b shows that validation accuracy increases rapidly at the first 20 epochs. Then, it increases slowly until around the fortieth epoch. It tends to be stable with a little fluctuation from 41 to 200 epochs. Convergence is achieved when the epoch is at around 121. The validation accuracy of the model does not increase despite more training iterations.

For Inception-ResNet-v2, Figure 10c presents that the validation accuracy increases rapidly at the first seventeenth epochs. It increases slowly until around forty-fourth epoch and tends to stable with a little fluctuation from 45 to 200 epochs. Convergence is achieved when it enters the epoch of 49. This shows that the Inception-ResNet-v2 model is faster to convergence than Xception.

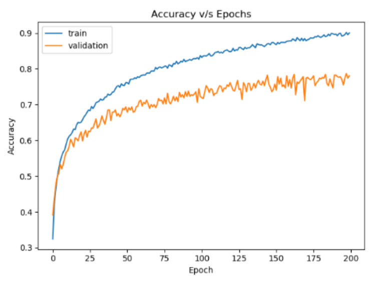
The learning rate serves to control how fast a model in solving problems. The larger levels of learning rates produce rapid changes and require less training epoch. Meanwhile, the smaller levels of learning rates require more training epoch because the smaller changes are made for the weight of each update. The level of learning rates that are too small can cause the optimization process to trap at a local minimum. Learning rates that are too large can cause the model to converge too quickly to sub-optimal solutions. The learning rate of 10^{-5} and 10^{-4} proves to be the right choice as a hyperparameter. It controls how much the change in the model is in response to the experiment.

Tables III–VI show that for all data classes, the use of 10^{-5} learning rate gets the highest accuracy of 31.48% of the 108 experimental data. Meanwhile, the use of 10^{-4} learning gets the highest accuracy of 68.52%. The data has shown that for this case, the optimal learning rate is achieved by using 10^{-4}. From the researchers' observation, RMSProp with learning rate 10^{-4} is more sensitive than 10^{-5}. This is proven by 62.96% of accuracy with 10^{-4} as it is higher compared to 37.04% of accuracy with 10^{-5}. If the researchers use Adam optimizer, 74.07% of accuracy with 10^{-4} is higher than 25.93% of accuracy with 10^{-5}. This result shows that by using the RMSProp optimizer or Adam optimizer, the learning rate of 10^{-4} results is more sensitive than 10^{-5}.

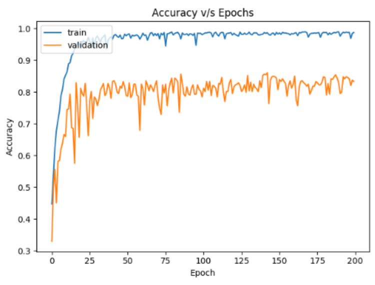
Figure 11 shows the progress of the accuracy of three classes of Inception-ResNet-v2 architecture with the same parameters (Adam optimizer, 200 epochs, and learning rate = 10^{-4}). The increase in the task with the addition of tea disease classes starts from two classes until final four classes. It shows that the validation value decreases along with the increase in the given task. Thus, the accuracy value decreases. The data sample used in this research is unbalanced (26.40% of healthy leaf image, 18.81% of blister blight image, 36.56% of *Empoasca sp.* (leafhoppers) image, and 19.23% of looper caterpillar image). The unbalanced classes make the validation difficult to represent the classes. Therefore, the researchers do not get optimized results.

V. CONCLUSION

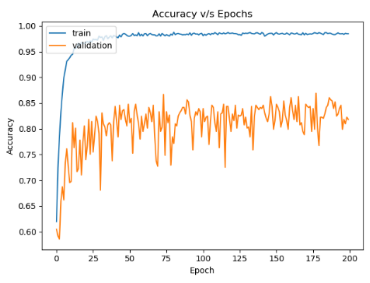
In this work, the concatenated CNN model is used for classifying the image of tea plant diseases. GoogLeNet, Xception, and Inception-ResNet-v2 architecture are used as a pre-trained model for classification. By varying optimizer, epoch, and learning rate, the model can achieve high recognition accuracy. The classification results indicate that the concatenated CNN can accurately classify the four types of tea



(a)
GoogLeNet architecture with 4 classes, accuracy: 78.65%, optimizer: RMSProp, batch: 10, epoch: 200, and learning rate: 10^{-5}

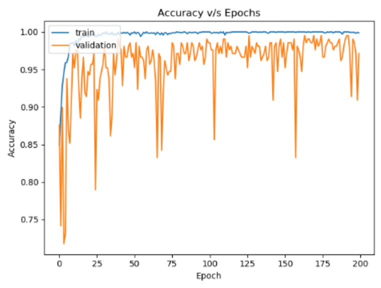


(b)
Xception architecture with 4 classes, accuracy: 87.10%, optimizer: RMSProp, batch: 10, epoch: 200, and learning rate: 10^{-4}

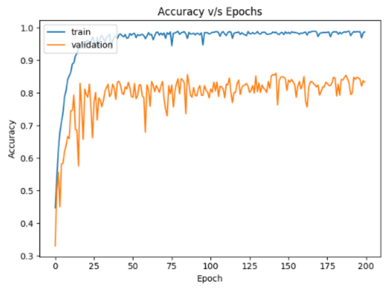


(c)
Inception-ResNet-v2 architecture with 4 classes, accuracy: 87.95%, optimizer: RMSProp, batch: 10, epoch: 200, and learning rate: 10^{-4}

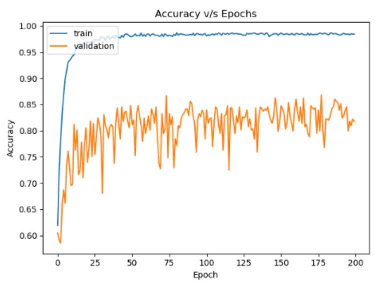
Fig. 10. The progression of the accuracy of the GoogLeNet, Xception, and Inception-ResNet-v2 with four classes and 200 epochs.



(a)
Inception-ResNet-v2 with 2 classes, accuracy: 97.13%, optimizer: Adam, batch: 10, epoch: 200, and learning rate: 10^{-4}



(b)
Inception-ResNet-v2 with 3 classes, accuracy: 90.05%, optimizer: Adam, batch: 10, epoch: 200, and learning rate: 10^{-4}



(c)
Inception-ResNet-v2 with 4 classes, accuracy: 87.95%, optimizer: Adam, batch: 10, epoch: 200, and learning rate: 10^{-4}

Fig. 11. The progression of the accuracy of the Inception-ResNet-v2 with 2, 3, and 4 classes, Adam optimizer, 200 epochs, and learning rate = 10^{-4}

plant diseases. It is with above 80% accuracy by using Inception-ResNet-v2 architecture, Adam optimizer, and 200 epochs.

Even though the results are good, the proposed method is still constrained by the length of training time. For forthcoming work, to speed up the training process and boost accuracy, future researchers can use batch normalization. They can also adjust the parameter to train well. To improve accuracy, the dataset can be reproduced and made balance.

Acknowledgement

This work is supported by The Ministry of Research, Technology and Higher Education of the Republic of Indonesia, and Research Center for Informatics, Indonesian Institute of Sciences 'Lembaga Ilmu Pengetahuan Indonesia' (LIPI). The computation in this research has been done using the facilities of High Performance Computer (HPC) in LIPI.

References

[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[2] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-Up Robust Features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[3] D. G. Tsolakidis, D. I. Kosmopoulos, and G. Papadourakis, "Plant leaf recognition using Zernike moments and Histogram of Oriented Gradients," in *Hellenic Conference on Artificial Intelligence*. Ioannina, Greece: Springer, May 15–17, 2014, pp. 406–417.

[4] M. S. Hossain, R. M. Mou, M. M. Hasan, S. Chakraborty, and M. A. Razzak, "Recognition and detection of tea leaf's diseases using Support Vector Machine," in *2018 IEEE 14th International Colloquium on Signal Processing & Its*

Applications (CSPA). Batu Feringghi, Malaysia: IEEE, March 9–10, 2018, pp. 150–154.

[5] R. Muralidharan and C. Chandrasekar, "Object recognition using SVM-KNN based on geometric moment invariant," *International Journal of Computer Trends and Technology*, vol. 1, no. 1, pp. 215–220, 2011.

[6] ——, "Object recognition using Support Vector Machine augmented by RST invariants," *International Journal of Computer Science Issues (IJCSI)*, vol. 8, no. 5, p. 280, 2011.

[7] Z. Q. Zhao, P. Zheng, S. T. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2019.

[8] S. Sladojevic, M. Arsenovic, A. Anderla, D. Culibrk, and D. Stefanovic, "Deep Neural Networks based recognition of plant diseases by leaf image classification," *Computational Intelligence and Neuroscience*, vol. 2016, pp. 1–11, 2016.

[9] S. Zhang, X. Wu, Z. You, and L. Zhang, "Leaf image based cucumber disease recognition using sparse representation classification," *Computers and Electronics in Agriculture*, vol. 134, no. March, pp. 135–141, 2017.

[10] A. K. Rangarajan, R. Purushothaman, and A. Ramesh, "Tomato crop disease classification using pre-trained deep learning algorithm," *Procedia Computer Science*, vol. 133, pp. 1040–1047, 2018.

[11] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," *Frontiers in Plant Science*, vol. 7, pp. 1–10, 2016.

[12] A. Ramcharan, P. McCloskey, K. Baranowski, N. Mbilinyi, L. Mrisho, M. Ndalahwa, J. Legg, and D. P. Hughes, "A mobile-based deep learning model for cassava disease diagnosis," *Frontiers in Plant Science*, vol. 10, pp. 1–8, 2019.

[13] B. Liu, Y. Zhang, D. He, and Y. Li, "Identification of apple leaf diseases based on Deep Convolutional Neural Networks," *Symmetry*, vol. 10, no. 1, p. 11, 2018.

[14] G. Geetharamani and A. Pandian, "Identification of plant leaf diseases using a nine-layer deep Convolutional Neural Network," *Computers & Electrical Engineering*, vol. 76, pp. 323–338, 2019.

[15] K. P. Ferentinos, "Deep learning models for plant disease detection and diagnosis," *Computers and Electronics in Agriculture*, vol. 145, pp. 311–318, 2018.

[16] A. T. Sapkal and U. V. Kulkarni, "Comparative study of leaf disease diagnosis system using texture features and deep learning features," *International Journal of Applied Engineering Research*, vol. 13, no. 19, pp. 14 334–14 340, 2018.

[17] P. M. Mainkar, S. Ghorpade, and M. Adawadkar, "Plant leaf disease detection and classification using image processing techniques," *International Journal of Innovative and Emerging Research in Engineering*, vol. 2, no. 4, pp. 139–144, 2015.

[18] J. G. Barbedo, "Factors influencing the use of deep learning for plant disease recognition," *Biosystems Engineering*, vol. 172, pp. 84–91, 2018.

[19] S. H. Lee, C. S. Chan, S. J. Mayo, and P. Remagnino, "How deep learning extracts and learns leaf features for plant classification," *Pattern Recognition*, vol. 71, pp. 1–13, 2017.

[20] M. Dyrmann, H. Karstoft, and H. S. Midtiby, "Plant species classification using Deep Convolutional Neural Network," *Biosystems Engineering*, vol. 151, pp. 72–80, 2016.

[21] X. Hao, W. Zhang, F. Zhao, Y. Liu, W. Qian, Y. Wang, L. Wang, J. Zeng, Y. Yang, and X. Wang, "Discovery of plant viruses from tea plant (Camellia sinensis (l.) o. kuntze) by metagenomic sequencing," *Frontiers in microbiology*, vol. 9, pp. 1–15, 2018.

[22] L. Keith, W. H. Ko, and D. M. Sato. (2006) Identification guide for diseases of tea (Camellia sinensis). [Online]. Available: https://scholarspace.manoa.hawaii.edu/bitstream/10125/12400/PD-33.pdf

[23] J. Chen, Q. Liu, and L. Gao, "Visual tea leaf disease recognition using a Convolutional Neural Network model," *Symmetry*, vol. 11, no. 3, p. 343, 2019.

[24] B. C. Karmokar, M. S. Ullah, M. K. Siddiquee, and K. M. R. Alam, "Tea leaf diseases recognition using neural network ensemble," *International Journal of Computer Applications*, vol. 114, no. 17, pp. 27–30, 2015.

[25] ImageNet. ImageNet Large Scale Visual Recognition Challenge (ILSVRC). [Online]. Available: http://www.image-net.org/challenges/LSVRC/

[26] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Boston, MA, June 7–12, 2015, pp. 1–9.

[27] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE Conference on Computer Vision and*

Pattern Recognition, Honolulu, Hawaii, United States, July 21–26, 2017, pp. 1251–1258.

[28] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, Nevada, United States, June 26–July 1, 2016, pp. 770–778.

[29] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “Inception-v4, inception-ResNet and the impact of residual connections on learning,” in *Thirty-First AAAI Conference on Artificial Intelligence*, San Francisco, California, USA, Feb. 04–09, 2017.

[30] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” in *3rd International Conference on Learning Representations (ICLR 2015)*, San Diego, CA, USA, May 7–9, 2015.