Classification of Rice Leaf Diseases from Digital Images using Convolutional Neural Network

Monica Widiasri
Informatics Engineering
Universitas Surabaya
Surabaya, Indonesia
monica@staff.ubaya.ac.id

Joko Siswantoro
Informatics Engineering
Universitas Surabaya
Surabaya, Indonesia
joko_siswantoro@staff.ubaya.ac.id

Putu Pramodia Suka Pratama
Informatics Engineering
Universitas Surabaya
Surabaya, Indonesia
pramodia32@gmail.com

Abstract*— Rice (Oryza sativa L.) is one of the main food crops in Indonesia. The Indonesian people rely on rice as their primary source of carbohydrates, so the quality and health of rice during the planting process must be considered to minimize the risk of crop failure. One of the causes of rice crop failure is the attack of pests or pathogens that attack rice. Pests or pathogens that attack rice plants cause diseases with a similar visual appearance, making it difficult for farmers to distinguish them. The purpose of this research is to create a convolutional neural network (CNN) model that is able to classify diseases on rice leaves from digital images to help farmers in distinguishing the visual characteristics of rice leaf diseases. Digital image data of five types of rice leaf diseases, namely bacterial leaf blight, blast, brown spot, leaf smut, and tungro were collected from three different sources. The CNN model used is a custom CNN model, which was evaluated with hyperparameter tuning and k-fold cross-validation. The best model was then retrained with a data ratio of 70:15:15 (training:validation:test) and obtained an accuracy metric equal to 0.9870; average precision equal to 0.9869; average recall equal to 0.9872; and average f1 score equal to 0.9870. The model was then implemented into an Android application using TensorFlow Lite. Based on the survey results, respondents agreed that the application can make it easier for farmers to classify the types of rice leaf diseases based on visual displays using digital images of rice leaf diseases.

Keywords*— Classification, Rice Leaf Disease, Convolutional neural network, K-Fold cross-validation, Android Application

I. INTRODUCTION

Rice (*Oryza sativa L.*) is one of the main food crops in Indonesia. The rice produced by rice plants is a major commodity essential to the Indonesian people and other Asian countries [1]. Indonesian society relies on rice as one of the primary sources of carbohydrates in its daily lives. This necessitates careful attention to the quality and health of rice during the planting process to prevent crop failure [2].

Rice production levels are greatly influenced by the health of the rice plants. A decline in production levels can be caused by several factors, one of which is pest infestation. Pests, or plant pests, are organisms that attack plants and can cause physiological damage. These organisms can also be referred to as pathogens due to their actions that can harm plants. Rice is a crop that pathogens can attack, and one of the parts of the rice plant that is often targeted is the rice leaf [3]. Some common diseases found on rice leaves include bacterial leaf blight, leaf smut, and brown spot. These three diseases have similar visual appearances, so farmers sometimes have difficulty distinguishing between them [4]. In addition to these three diseases, there are also tungro and blast diseases that frequently attack rice leaves [5].

Convolutional neural networks (CNNs) are a type of neural network architecture used to recognize visual patterns directly from pixel images with varying levels of complexity. CNNs are also a special form of feed-forward neural networks (FNNs) trained with back-propagation, also known as multi-layer perceptrons (MLPs) [6]. CNN combines three architectural ideas to ensure shift, scale, and distortion invariance, namely local receptive fields, shared weights, and spatial or temporal subsampling [7]. The CNN model was chosen to solve this problem because it can extract basic features from an image and combine these features to detect higher-level features, such as objects, faces, or more complex patterns. Building on previous research that has employed various machine learning methods to classify rice diseases, this study selected the CNN model because it has been shown to perform well in classifying rice diseases. This classification process must be carried out as quickly as possible and be readily available to prevent the further spread of rice diseases. This need requires CNN models to be as efficient and flexible as possible for farmers. According to a survey cited in reference [8], various Android applications have been developed to meet the needs of farmers. Approximately 12% of these agricultural applications focus on controlling agricultural pests and diseases. Additionally, nearly all segments of society in the modern era have smartphones capable of running applications [9]. Based on these reasons, Android applications were selected as the platform for the CNN model to be developed.

The implementation of CNN models into Android applications has been done by researchers to solve other classification problems, such as COVID-19 detection from X-ray images using CNN and Android applications[10]. This was done using TensorFlow Lite. TensorFlow Lite is a set of tools that allows developers to run their machine learning models on mobile devices. The main feature of TensorFlow Lite is its ability to convert TensorFlow models into TensorFlow Lite Models and its low latency, which improves model performance speed. Additionally, TensorFlow Lite is easy to use because it offers a relatively simple way to develop machine learning models on Android applications. Another important feature of TensorFlow Lite is that it can be used without an internet connection, enabling developers to implement machine learning models in remote locations without internet access[11]. Based on the above key features, TensorFlow Lite was selected as the tool to assist in the implementation of the CNN model into the Android application.

II. RESEARCH METHOD

This research comprises five main stages: dataset collection, model design and hyperparameter tuning, training, model evaluation, and classification process on the Android application.

A. Dataset Collection

The datasets used can be divided into three types based on their sources, namely datasets from the UCI Machine Learning Repository, Mendeley Data, and datasets from other researchers. The three datasets were combined into a new directory, containing a total of 6,302 images. The classes selected from each dataset were rice leaf disease classes relevant to the scope, namely bacterial leaf blight, blast, brown spot, leaf smut, and tungro, as shown in Figure 1.

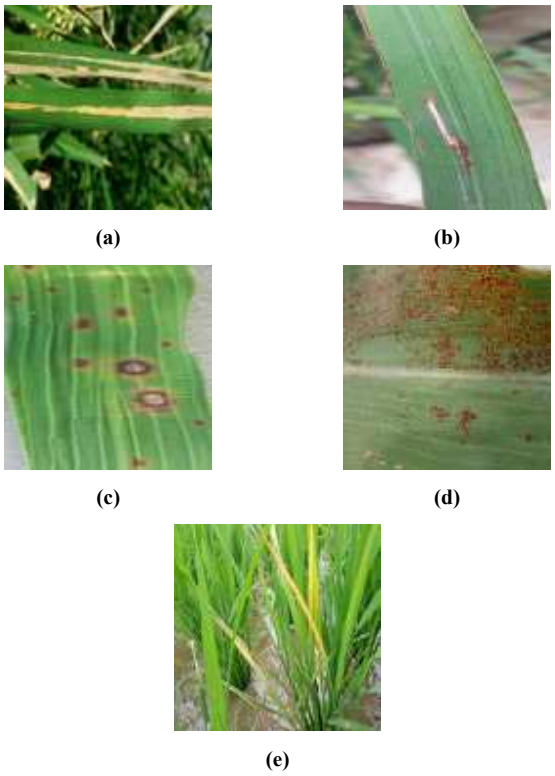

Fig. 1. Images of Rice Leaf Diseases, (a) Bacterial leaf blight, (b) Blast, (c) Brown spot, (d) Leaf smut, and (e) Tungro

B. Model Design and Hyperparameter Tuning

The VGG16 model was modified to classify pneumonia medical images, as VGG-16 had poor performance in classifying medical images [12]. The poor performance of VGG-16 because the original model was trained using 1 million images from ImageNet with 1,000 classes, so its application to a small dataset can lead to underfitting. The model used is called IVGG13, which reduces the network depth compared to VGG-16 to avoid underfitting and overfitting during training. Jiang et al. demonstrate that the IVGG13 model requires less training time and computational resources than the VGG-16 model [12]. IVGG13 reduces layer depth with small convolution kernel sizes, thereby significantly reducing the number of network parameters.

Based on the results obtained from previous studies, this research proposes a modified VGG-16 model to reduce network depth (custom model). The modifications include reducing the number of hidden and convolutional layers, as well as the number of filters in each convolutional and fully connected layer.

Hyperparameter tuning was performed to find the number of hidden layer blocks, the number of convolutional layers in each hidden layer block, the best kernel size value for each convolution layer, the best number of units in each convolution layer and dense layer, and the best dropout rate value in the dropout layer. Hyperparameter search was performed using the random search technique. The list of hyperparameters whose values were searched using the random search process is presented in Table I.

TABLE I HYPERPARAMETER LIST

Hyperparameter	Description	Search Space
Number of blocks	Number of hidden layer blocks.	[1,2,3,4,5]
Number of convolution layer	The number of convolution layers in each block.	[1,2]
Unit (convolution layer and dense layer)	Number of units/filters in each convolution layer (5 layers).	[8,16,32,64,64], [16,32,64,128,128], [32,64,128,256,256]
	Number of units/filters in the first two dense layers.	[64,64], [128,128], [256,256]
	Combination of convolution unit/filter and dense layer.	[8,16,32,64,64,64,64], [16,32,64,128,128,128, 128], [32,64,128,256,256,25 6,256]
Kernel size	The kernel size in each convolution layer.	[3,5]
Dropout rate	The probability of ignoring a number of neurons during training	[0.1,0.2,0.3,0.4,0.5]

C. Model Training Process

The training process aims to prepare a classification model that will be used for evaluation. This process involves hyperparameter tuning to determine the optimal hyperparameters for the model, followed by retraining the model with these optimized hyperparameters. The training process can be seen in Figure 2.

The first step is to create a dataframe from the dataset. This dataframe consists of two columns, namely the path and class columns. This is done because the K-fold cross-validation process requires the flow_from_dataframe() method from the ImageDataGenerator class, which requires a dataframe as its main argument. Next, the order of the dataframe contents is randomized. This is done to ensure that the data is not sorted by class, which could cause class imbalance during the K-fold cross-validation process with K=5.

The next step is to perform hyperparameter tuning using the random search method with 20 trials. Each trial will train and evaluate the model with a random combination of hyperparameters. The model is evaluated using K-fold cross-validation. The selected model is the one with the highest accuracy value.

Before the final stage of the training process, data augmentation techniques are employed to address class imbalance issues, utilizing rotation, width shift, height shift, shear, zoom, horizontal flip, and vertical flip. Using the augmented dataset, the final stage, model retraining, was performed. The model selected at this stage is the model with the best combination of hyperparameter values. The model is retrained using 70% of the dataset as training data and 15%

as validation data. The result of this process is a model that is ready for evaluation using test data.

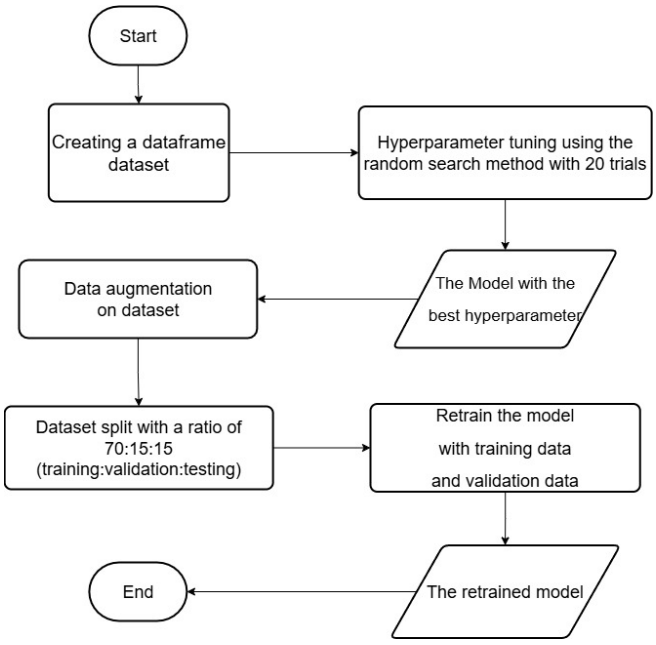

Fig. 2. Training Process Flowchart

.

D. *Classification Process in Android Applications*

The process that occurs in the Android application covers all processes from image capture by the user to displaying the classification results. The model used at this stage is the TFLite model from the model evaluation stage.

The first process is image capture by the user. Users can capture images directly using the device's camera or take pictures from the gallery. The second process is image preprocessing, which resizes and rescales the images used. The images will be resized to 224 × 224 pixels, and the pixel intensity range will be changed to 0 to 1. The third process is image classification by the TFLite model. The probability results generated by the model are then compared with the threshold value. The final process is displaying the classification results. If the probability value is less than the threshold, the system will display information that rice leaf disease was not found. Meanwhile, if the probability value is greater than or equal to the threshold, the system will display all information about the classified rice leaves.

III. RESULT

The results of this study include the model architecture with the best hyperparameters, the best model re-evaluation values, and the outcomes of implementing the user interface on the Android application. The results of hyperparameter tuning can be seen in Tables II and III.

The custom model with hyperparameter configuration in trial 20 produced the best metric values. Not only did it have the best accuracy value, but this model also produced the best recall and F1 score values. This model was also retrained using 70% of the dataset for the training set and 15% of the dataset for the validation set, and then evaluated using 15% of the dataset for the test set. To tackle the imbalanced class problem, some of the data in the dataset were augmented using a data augmentation technique. The training process stopped at epoch 22. The selected model architecture is illustrated in Figure 3. For comparison, the model was also evaluated against the IVGG13 and VGG16 models. Tables IV and V show a comparison of the models' accuracy values without and with data augmentation, respectively.

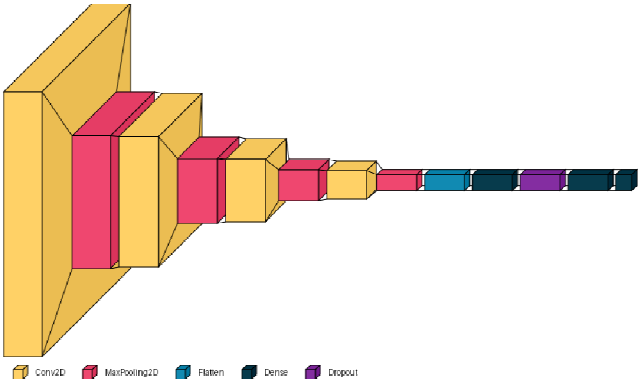

Fig. 3 Selected Model Architecture

TABLE II LIST OF HYPERPARAMETERS ON EACH TRIAL

Trial	Hyperparameter				
	units	*kernel_ size*	*block _size*	*layer_ count*	*drop out*
1	[8, 16, 32, 64, 64, 64, 64]	3	3	1	0.4
2	[16, 32, 64, 128, 128, 128, 128]	5	4	1	0.1
3	[16, 32, 64, 128, 128, 128, 128]	3	4	2	0.5
4	[8, 16, 32, 64, 64, 64, 64]	5	3	1	0.5
5	[8, 16, 32, 64, 64, 64, 64]	5	1	1	0.1
6	[32, 64, 128, 256, 256, 256, 256]	3	4	1	0.4
7	[32, 64, 128, 256, 256, 256, 256]	3	5	1	0.4
8	[16, 32, 64, 128, 128, 128, 128]	3	3	1	0.2
9	[16, 32, 64, 128, 128, 128, 128]	5	1	2	0.5
10	[32, 64, 128, 256, 256, 256, 256]	3	2	2	0.1
11	[32, 64, 128, 256, 256, 256, 256]	3	3	2	0.5
12	[16, 32, 64, 128, 128, 128, 128]	5	5	1	0.4
13	[8, 16, 32, 64, 64, 64, 64]	5	4	1	0.3
14	[32, 64, 128, 256, 256, 256, 256]	5	1	2	0.5
15	[8, 16, 32, 64, 64, 64, 64]	3	5	2	0.3
16	[16, 32, 64, 128, 128, 128, 128]	3	4	1	0.3
17	[32, 64, 128, 256, 256, 256, 256]	5	2	1	0.5
18	[8, 16, 32, 64, 64, 64, 64]	3	2	1	0.4
19	[16, 32, 64, 128, 128, 128, 128]	3	4	1	0.5
20	[32, 64, 128, 256, 256, 256, 256]	3	4	1	0.4

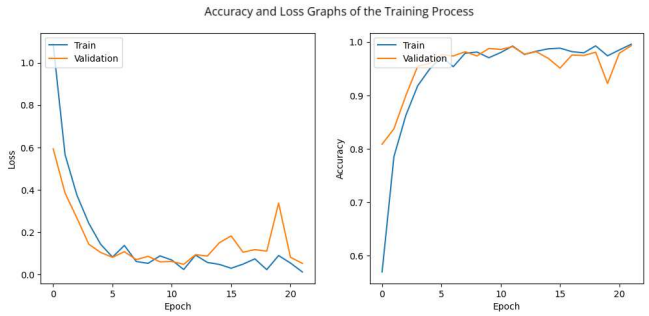

Fig. 4. Accuracy And Loss Graphs of the Training Process

The model evaluation process also produced precision, recall, and F1 scores for each class of rice leaf disease. These metric values were obtained when evaluating the custom model, IVGG13, and VGG16 with 15% of the test data. Tables VI and VII present the precision, recall, and F1 scores for each class in the custom model, IVGG13, and VGG16 respectively, without and with data augmentation. Based on the evaluation results of the custom model, IVGG13, and VGG16, it was found that the custom model had the best accuracy, precision, recall, and F1 scores.

TABLE III HYPERPARAMETER TUNING RESULT

Trial	Metrics			
	Accuracy	*Precision*	*Recall*	*F1 Score*
1	0.9711	0.9733	0.9676	0.9218
2	0.9381	0.9447	0.9310	0.8692
3	0.8499	0.9056	0.7775	0.7176
4	0.9621	0.9703	0.9568	0.8724
5	0.9151	0.9339	0.9030	0.7970
6	0.9710	0.9737	0.9691	0.9199
7	0.9675	0.9711	0.9641	0.9086
8	0.9643	0.9666	0.9614	0.9072
9	0.8810	0.9014	0.8662	0.7899
10	0.9165	0.9348	0.9077	0.8438
11	0.8629	0.8944	0.8296	0.7473
12	0.9719	0.9749	0.9676	0.9047
13	0.9602	0.9656	0.9560	0.8861
14	0.6938	0.7061	0.6805	0.5771
15	0.8816	0.9075	0.8543	0.7886
16	0.9679	0.9711	0.9656	0.9078
17	0.9356	0.9505	0.9262	0.8346
18	0.9538	0.9613	0.9446	0.8358
19	0.9733	0.9764	0.9700	0.9201
20	**0.9737**	**0.9759**	**0.9703**	**0.9237**

The accuracy and loss graphs during the custom model training process are shown in Figure 4. These graphs show that the model does not experience excessive overfitting. This also indicates that the model selection process, achieved through hyperparameter tuning using the K-fold cross-validation method, can produce a model that does not experience excessive overfitting.

The confusion matrix from the evaluation results can be seen in Figure 5. According to the confusion matrix results, it is evident that the model accurately predicts five types of rice leaf diseases. Based on the model evaluation results, the custom model with hyperparameter configuration in trial 20 was implemented into the Android application.

System validation was conducted by testing the Android application on 50 rice farmers. Afterward, respondents completed a questionnaire to assess their experience using the rice leaf disease classification application. In this questionnaire, the farmers were also asked to classify a set of rice leaf diseases based on their experience and knowledge. The same image was also classified by application. The results of implementing the best model into the Android application can also be seen in Figure 6.

TABLE IV MODEL EVALUATION RESULT WITHOUT DATA AUGMENTATION

Model	Accuracy		Average Precision		Average Recall		Average F1 Score	
	Val.	*Test*	*Val.*	*Test*	*Val.*	*Test*	*Val.*	*Test*
Custom	**0.9745**	**0.9663**	**0.9213**	**0.9353**	**0.9214**	**0.9358**	**0.9214**	**0.9355**
IVGG13	0.9311	0.9200	0.8489	0.8682	0.8855	0.8843	0.8625	0.8756
VGG16	0.2153	0.2158	0.0431	0.0432	0.2000	0.2000	0.0709	0.0710

TABLE V MODEL EVALUATION RESULT WITH DATA AUGMENTATION

Model	Accuracy		Average Precision		Average Recall		Average F1 Score	
	Val.	*Test*	*Val.*	*Test*	*Val.*	*Test*	*Val.*	*Test*
Custom	**0.9913**	**0.9870**	**0.9914**	**0.9869**	**0.9910**	**0.9872**	**0.9912**	**0.9870**
IVGG13	0.9318	0.9235	0.9320	0.9238	0.9325	0.9243	0.9316	0.9233
VGG16	0.2194	0.2189	0.0439	0.0438	0.2000	0.2000	0.0720	0.0718

IV. DISCUSSION

The hyperparameter tuning results were obtained by performing k-fold cross-validation on each trial. From these results, it was found that the highest accuracy was achieved on the 20th trial with the following unit types: [32,64,128,256,256,256,256], kernel size 3 × 3, block size 4, with 1 convolutional layer per block, and dropout rate of 0.4. The optimal architecture used in this study (custom model) was obtained with 32, 64, 128, and 256 filters in the first four convolutional layers (the fifth layer was not used), followed by two dense layers consisting of 256 units each, as shown in Figure 3.

Based on the experimental results obtained, the custom model with the hyperparameter configuration proposed in this study successfully outperformed both IVGG13 and VGG-16 models. Hyperparameter configuration using the random search technique successfully reduced network depth and found model parameter values, resulting in the best classification performance.

The model with the best hyperparameters was retrained using a dataset ratio of 70% for training and 15% for validation, and then evaluated using 15% of the test data. Data augmentation technique was also used in this dataset to tackle the class imbalance problem, and it improved the model performance. The evaluation results indicate that the model's performance decreased when evaluated using the validation and test sets, although the decrease was not significant. The model selection process, which involved hyperparameter tuning using the k-fold cross-validation method, also produced a model that did not experience significant overfitting, as shown in Figure 4.

TABLE VI PRECISION, RECALL, AND F1 SCORE VALUES WITHOUT DATA AUGMENTATION

Class	Precision			Recall			F1 Score		
	Custom	*IVGG13*	*VGG16*	*Custom*	*IVGG13*	*VGG16*	*Custom*	*IVGG13*	*VGG16*
Bacterial leaf blight	0.9719	0.9502	0.0000	0.9603	0.9087	0.0000	0.9661	0.9290	0.0000
Blast	0.9552	0.8874	0.0000	0.9509	0.9152	0.0000	0.9530	0.9011	0.0000
Brown spot	0.9686	0.9492	0.0000	0.9724	0.9567	0.0000	0.9705	0.9529	0.0000
Leaf smut	0.8000	0.6471	0.0000	0.8000	0.7333	0.0000	0.8000	0.6875	0.0000
Tungro	0.9808	0.9073	0.2158	0.9951	0.9073	1.0000	0.9879	0.9073	0.3550

TABLE VII PRECISION, RECALL, AND F1 SCORE VALUES WITH DATA AUGMENTATION

Class	Precision			Recall			F1 Score		
	Custom	*IVGG13*	*VGG16*	*Custom*	*IVGG13*	*VGG16*	*Custom*	*IVGG13*	*VGG16*
Bacterial leaf blight	0.9843	0.9365	0.2189	0.9960	0.9365	1.0000	0.9901	0.9365	0.3592
Blast	0.9955	0.9455	0.0000	0.9821	0.8527	0.0000	0.9888	0.8967	0.0000
Brown spot	0.9920	0.9209	0.0000	0.9764	0.9173	0.0000	0.9841	0.9191	0.0000
Leaf smut	0.9726	0.8947	0.0000	0.9861	0.9444	0.0000	0.9793	0.9189	0.0000
Tungro	0.9903	0.9213	0.0000	0.9951	0.9707	0.0000	0.9927	0.9454	0.0000

Based on the rice leaf disease classification results, the proposed custom model performed very well in classifying all five diseases, with classification performance (measured by precision, recall, and F1 score) exceeding 97%. The data augmentation process also enhances classification performance, as indicated by an improvement in the model's ability to classify leaf smut from 80% to 97%.

The validation results of the questionnaire produced an average score of 2.26 on a scale of 3. This indicates that the Android application for classifying rice leaf diseases based on visual displays was well-received. Based on the results of the questionnaire, the value of the farmer's accuracy in classifying rice leaf disease is 34.8% and the accuracy of the application in classifying the same images is 91.2%. This indicates the application can classify rice leaf disease more accurately. Detailed information can be found in Table VIII.

TABLE VIII FARMER AND APPLICATION ACCURACY

Class	Farmer		Application	
	True	*False*	*True*	*False*
Bacterial Leaf Blight	18	32	43	7
Blast	5	45	49	1
Brown Spot	15	35	44	6
Leaf Smut	17	33	49	1
Tungro	32	18	43	7
Accuracy	**0.348**		**0.912**	

TABLE IX QUESTIONNAIRE RESULT

Question	Disagree	Agree	Strongly Agree
Is this application easy to use for identifying diseases in rice leaves?	0	43	7
Are the instructions or layout of the application easy to understand?	0	43	7
Are the classification results produced by the system correct?	0	14	36
Are the classification results easy to understand?	0	43	7
Do the detailed disease features help you understand the impact, causes, identification methods, and solutions for rice leaf disease?	0	42	8
Total Score	**565**		
Average Score	**2.26**		

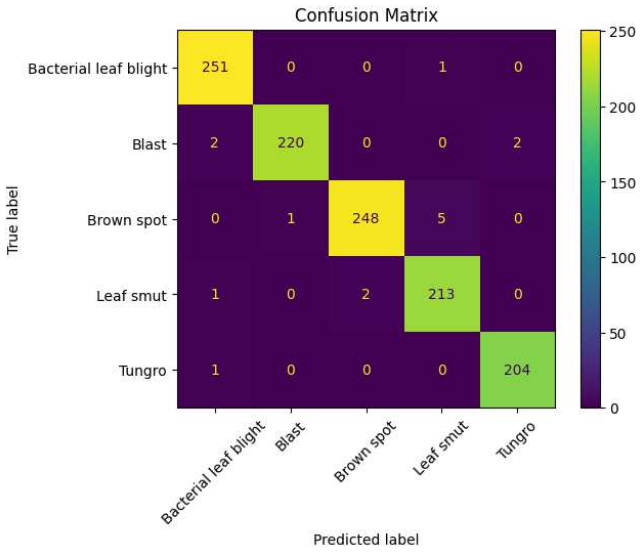

Fig. 5. Confusion Matrix from Model Evaluation

Respondents agreed that this application could facilitate farmers in classifying rice leaf diseases based on visual displays using digital images of rice leaf diseases. The detailed questionnaire's result can be seen in Table IX.

V. CONCLUSION

Based on the research results, a CNN model that can classify diseases in rice leaves from digital images has been successfully implemented in an Android application. After retraining with the best model and evaluating it with the test set, the accuracy metric obtained was 0.9870; the average precision was 0.9869; the average recall was 0.9872; and the F1 score was 0.9870. The hyperparameter selection process using k-fold cross-validation also ensured that the selected model did not overfit. To improve classification results in future research, increase the number of trials in the hyperparameter tuning process using random search to find more optimal hyperparameter values.

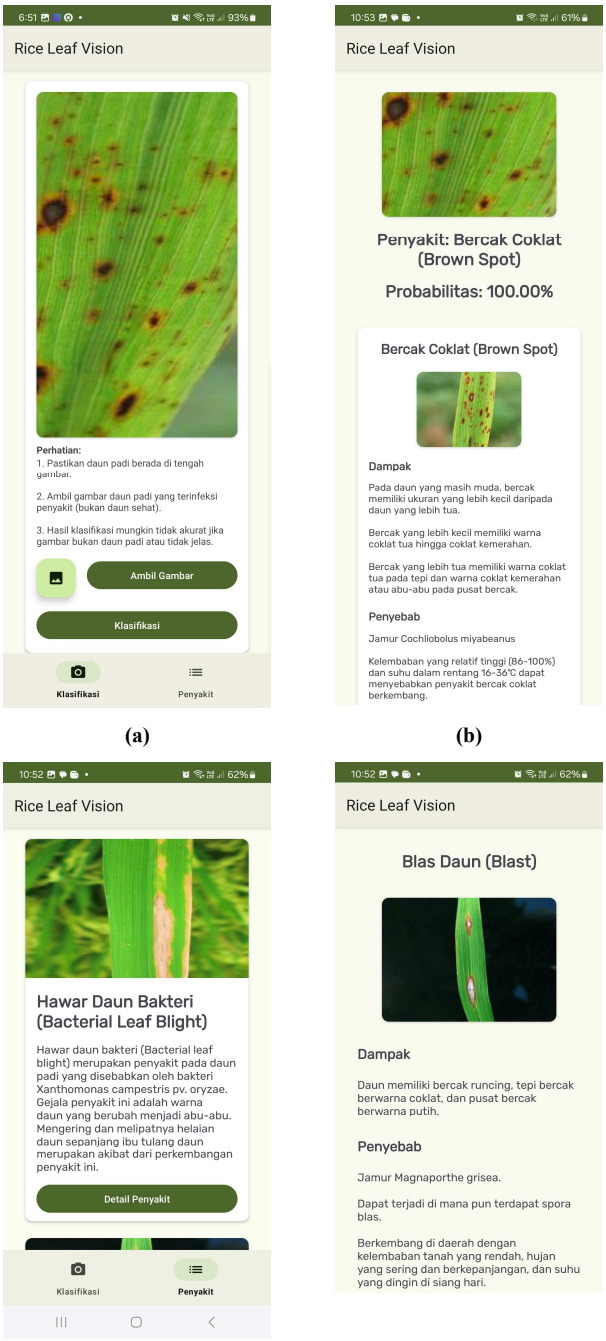

Fig. 6. Page Display, (a) Classification, (b) Classification Results, (c) List of Diseases, (d) Disease Details

REFERENCES

[1] R. A. Saputra, S. Wasiyanti, A. Supriyatna, and D. F. Saefudin, "Penerapan Algoritma Convolutional Neural Network Dan Arsitektur MobileNet Pada Aplikasi Deteksi Penyakit Daun Padi," *Swabumi*, vol. 9, no. 2, pp. 184–188, 2021, doi: 10.31294/swabumi.v9i2.11678.

[2] A. M. Alif *et al.*, "Pengaruh Media Tanam Sistem Irigasi Dalam Meningkatkan," vol. 7, pp. 2499–2505, 2023.

[3] A. Walascha, A. Febriana, D. Saputri, D. Sri Nur Haryanti, R. Tsania, and Y. Sanjaya, "Review Artikel: Inventarisasi Jenis Penyakit yang Menyerang Daun Tanaman Padi (Oryza sativa L.)," in *Prosiding Seminar Nasional Biologi*, 2022, vol. 1, no. 2, pp. 471–478.

[4] M. E. Pothen and D. M. L. Pai, "Detection of Rice Leaf Diseases Using Image Processing," in *Proceedings of the 4th International Conference on Computing Methodologies and Communication, ICCMC 2020*, 2020, no. Iccmc, pp. 424–430, doi: 10.1109/ICCMC48092.2020.ICCMC-00080.

[5] A. Julianto and A. Sunyoto, "A performance evaluation of convolutional neural network architecture for classification of rice leaf disease," *IAES Int. J. Artif. Intell.*, vol. 10, no. 4, pp. 1069–1078, 2021, doi: 10.11591/IJAI.V10.I4.PP1069-1078.

[6] C. C. J. Kuo, "Understanding convolutional neural networks with a mathematical model," *J. Vis. Commun. Image Represent.*, vol. 41, pp. 406–413, 2016, doi: 10.1016/j.jvcir.2016.11.003.

[7] Y. Lecun, L. Bottou, Y. Bengio, and P. Ha, "Gradient-Based Learning Applied to Document Recognition," *Proc. IEEE*, no. November, pp. 1–46, 1998.

[8] H. Patel and D. Patel, "Survey of Android Apps for Agriculture Sector," *Int. J. Inf. Sci. Tech.*, vol. 6, no. 1/2, pp. 61–67, 2016, doi: 10.5121/ijist.2016.6207.

[9] A. Budiman, P. Utomo, and S. Rahayu, "Pengembangan Aplikasi Deteksi Dini Serangan Hama Padi Berbasis Android," *J. Terap. Abdimas*, vol. 4, no. 1, p. 33, 2019, doi: 10.25273/jta.v4i1.3805.

[10] K. F. Bushra, M. A. Ahamed, and M. Ahmad, "Automated detection of COVID-19 from X-ray images using CNN and Android mobile," *Res. Biomed. Eng.*, vol. 37, no. 3, pp. 545–552, 2021, doi: 10.1007/s42600-021-00163-2.

[11] C. C. Mart´ın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen and and X. Z. Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Man´e, Rajat Monga, Sherry Moore, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," *Netw. Comput. Neural Syst.*, 2015, doi: 10.1080/09548980500300507.

[12] Z. P. Jiang, Y. Y. Liu, Z. E. Shao, and K. W. Huang, "An improved VGG16 model for pneumonia image classification," *Appl. Sci.*, vol. 11, no. 23, 2021, doi: 10.3390/app112311185.