International Journal of Electrical and Computer Engineering (IJECE)
Vol. 13, No. 3, June 2023, pp. 2908~2920
ISSN: 2088-8708, DOI: 10.11591/ijece.v13i3.pp2908-2920

Corn plant disease classification based on leaf using residual networks-9 architecture

Tegar Arifin Prasetyo[1], Victor Lambok Desrony[1], Henny Flora Panjaitan[1], Romauli Sianipar[1], Yohanssen Pratama[2]
[1]Department of Information Technology, Faculty of Vocational Studies, Institut Teknologi Del, Toba, Indonesia
[2]Department of Software Engineering Technology, Faculty of Vocational Studies, Institut Teknologi Del, Toba, Indonesia

Article Info

Article history:

Received Aug 5, 2022
Revised Sep 12, 2022
Accepted Oct 1, 2022

Keywords:

Convolutional neural network
Corn plant disease
Disease classification
Hyperparameter tuning
ResNet-9 architecture

ABSTRACT

Corn plants are classified based on the leaf as healthy leafy and have 3 types of diseases leaf namely northern leaf blight, common rust, and gray leaf spot. Convolutional neural network (CNN) is the most popular structure for classification image detection. In this study, ResNet-9 architecture was implemented to build the best model CNN for the classification of corn plant diseases. After that, we do comparisons of epochs 5, 25, 55, 75, and 100 to get the best model. The highest accuracy value was obtained in the 100-epoch experiment so in this study 100 epochs were used in model formation. The dataset source in this study uses a dataset taken from the Kaggle platform as many as 9145 leaf corn plant data which is divided into training data (80%) and testing data (80%). In this study, three hyperparameter tuning experiments were carried out and the results of hyperparameter tuning experiments where num_workers is 4 and batch_size is 32. This classification obtained an accuracy rate of 99% and the model is implemented into a web interface.

Corresponding Author:

Tegar Arifin Prasetyo
Department of Information Technology, Faculty of Vocational Studies, Institut Teknologi Del
Sisingamangaraja Street, Sitoluama, Laguboti, Toba, North Sumatra 22381, Indonesia
Email: tegar.prasetyo@del.ac.id

1. INTRODUCTION

Corn is a carbohydrate-producing plant that occupies the second position in Indonesia after rice [1]. From the data obtained in 2021, the amount of corn production in Indonesia reaches around 15.79 million tons, proving that corn production in Indonesia is very large [2]. Every plant will never be free from diseases that make plants unable to carry out their normal physiological growth properly [3], [4]. In general, disease in plants determines the viability of a plant to be produced. In corn plants, there are 3 types of common diseases, there are common rust, gray leaf spot, and northern leaf blight [5]–[9]. Common rust is a disease of corn plants with symptoms experienced by small round to oval spots on the leaf surface on the top and bottom [10]. Gray leaf spot is a disease with symptoms experienced in the form of greyish-brown spots on the entire leaf surface [11]. Northern leaf blight has symptoms in the form of small, oval-shaped spots, while the more elongated spots are grey or brown in color [12].

The application of classification in various areas of human life can help humans solve a problem [13], [14]. The use of artificial intelligence that works intelligently in computer vision provides useful information for humans through visual images such as animals, plants, and others which aims to provide basic information about the existence of an object and the classification of an object [15], [16]. In this case, the object to be detected in the research conducted is to use a case study of corn plants and the focus of the classification object is to identify leaf diseases found in corn plants. Diseases in maize are still difficult to

monitor manually because of the limited number and working time of farm workers. So that not a few mistakes are made in handling disease problems in corn plants.

In recent years, machine learning and deep learning research in the detection of image classification of plant diseases have become the center of research [17]. Several machine learning and deep learning research methods that have been successfully applied in corn plant disease classification mainly include support vector machine (SVM) [18], k-nearest neighbors (KNN) [19], and convolutional neural networks (CNN) [5]. Aravind *et al*. [18] used as many as 2,000 image data obtained from the PlantVillage open access image database and implemented an SVM to classify types of corn plant diseases, namely common rust, *Cercospora* leaf spot, leaf blight, and healthy, and achieve a best average accuracy is 83.7%. Athani *et al.* [19] used 500 image corn plant crops and implemented an SVM and achieve an accuracy of 82% which was determined by using k-fold cross-validation. Khotimah [20] used 200 data for the identification of maize plant nutrients by doing a comparative analysis of KNN and Naïve Bayes method and their results show that KNN classification performance is more accurate than Naïve Bayes in the average accuracy of KNN is 92.40% with the best k difference at $k = 7$. Hidayat *et al.* [5] used 3,854 image data to classify corn plant diseases are categorized into common rust, gray leaf spot, and northern leaf blight diseases and the accuracy result obtained is 99%. From the accuracy results obtained, it can be seen that the application of the CNN algorithm in data classification is high [5].

CNN is the most popular structure for classification image detection [21]–[24]. In our study, we use CNN because we want to get the best accuracy and CNN has been proven to be very effective in image disease plant classification research [25] and also CNN extracts features automatically [26], [27]. Here, we propose to use ResNet-9 architecture for building model CNN to classify disease corn plants. residual networks (ResNet) are a convolutional network that is trained on more than 1 million images from the ImageNet database [28]–[30] and ResNet-9 has a pretrained network that can classify images of up to 1,000 object categories which makes the network used to learn good feature representation for various images. The main contribution in this study can be highlighted as: i) designing of new CNN model for diseases classification on corn plants; ii) doing comparisons of epochs with the distribution of 80% training data and 20% testing data have been carried out to obtain the best model; iii) using ResNet-9 architecture and Adam optimizer to train the best model; and iv) building a web interface that can classify diseases in corn plants which will display the label and the accuracy of classification results displayed.

2. RESEARCH METHODS

The explanation of the data processing design scheme is as follows as shown in Figure 1.

a) Image data are used as input to be processed.
b) After the image data is input, image preprocessing is carried out first, where image resizing will be carried out which serves to select the image size according to the image to be scaled so that image distortion does not occur.

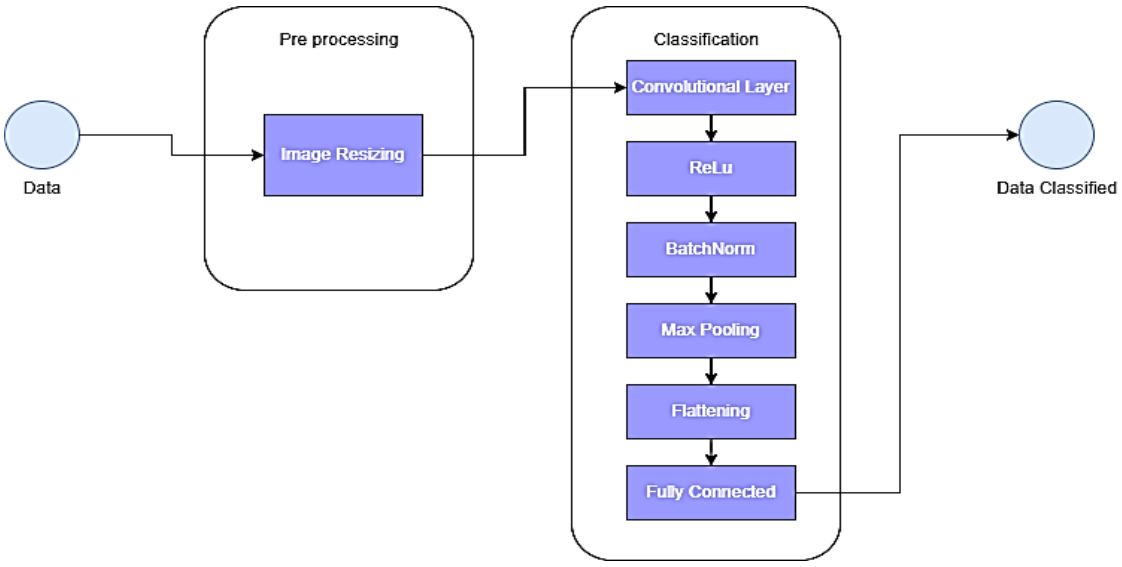


Figure 1. Data processing design scheme

c) Next, the classification stage will be carried out using the CNN algorithm, which consists of several layers, namely the convolution layer, rectification linear unit (ReLU), batch normalization, max pooling, flattening, and fully connected layer.
d) In the convolutional layer, the first thing to do is determine the pixel value that will be used to perform the convolution.
e) In ReLU, there will be units that increase the representation of a model and introduce nonlinearity, which in this layer will be followed by a convolutional layer.
f) Batch normalization is used to normalize the activation before passing it on to the next layer in the network.
g) In max pooling will choose the maximum value of all pixels where max pooling is located on the pooling layer which is the layer used for input and processing of various statistical activities based on the nearest pixel value.
h) In the flattening stage, the results of the pooled feature map are converted in the form of a 2-dimensional array into one long linear vector
i) In the fully connected layer stage, the layer taken from the input of the previous process will determine which features are most closely related to a particular class.
j) From the classification stages that have been carried out, classification data will be produced as output consisting of 4 classifications.

2.1. Data

The data to be used is taken from the dataset source that provides the image dataset. The source of the dataset in this study will use a dataset taken from the Kaggle platform. Data retrieval from the Kaggle dataset was carried out because the selected dataset presented data that was in accordance with the needs of this research. In collecting datasets, there are types of image criteria for training and testing in the CNN method [31], [32]. Data collection for training data was taken as many as 1,916 for common rust, 1,651 for *Cercospora* leaf spot gray leaf spot, 1907 for northern leaf blight, and for healthy there were 1858 images. As for testing data, 478 were taken for common rust, 411 for *Cercospora* leaf spot gray leaf spot, 478 for northern leaf blight, and for healthy there were 466 images, where the total data is 9165 data which is divided into two, namely training data (80%) as much as 7,332 data and testing data (20%) as much as 1,833 data. The characteristics of the training data consist of 4 parts, namely common rust, gray leaf spot, northern leaf blight, and healthy. In collecting the dataset, images will be collected from each leaf object. After the image data is collected, it will be grouped into folders to group leaf objects [33].

2.2. Image preprocessing

The data used in this study is that the images taken are clearly visible, based on the shooting requirements described in the dataset analysis subsection. Preprocessing has the goal that the dataset to be used is processed to produce the data needed by the system by removing unnecessary information from image data [34]. The preprocessing process has stages, namely image cutting, and image resizing.

The dataset of corn leaf images in this study obtained from the Kaggle platform consists of colored or red green blue (RGB) leaf images. The image will show which parts need to be cropped to remove unwanted information such as the background of the image or the presence of other images around the leaf image. The detected corn leaf image is cropped in such a way that the remaining image is only the leaf area that contains the parts required for leaf grouping. Cropping the image will help remove unnecessary parts of the image such as the background of the image, and other images that are around the image, but it is enough to only leave the part of the leaf image that is needed. The part of the leaf that will be detected can be in the form of an image of a whole leaf that has blades and leaf midribs, both front and back, which only focuses on the leaf image.

After obtaining the required portion of the given corn leaf image, it can be ensured that the cropped leaf image should not be too small or too large in order to be processed because the image must still be clearly visible. Therefore, it is necessary to select an appropriate image size for the image to be scaled to avoid image distortion. For the image used in this study, the leaf size will be cut to 256×256 pixels which only contains the leaf part of the original image without interference from other images around it.

2.3. Classification system block

The classification system block as shown in Figure 2 is the workflow of the classification system where the first process is the design CNN architecture model or designing the architecture that will be used. Then the second process is to do image preprocessing or a process that will change the image size according to the CNN architectural configuration that will be used and that has been designed. The third process is the CNN train architecture model or conducting training and validation on pre-processed datasets. Furthermore,

CNN architectural model testing is a process that will test the CNN architectural model that has been trained and tested. The test results will be stored and will be used for evaluation and conclusions at a later stage.

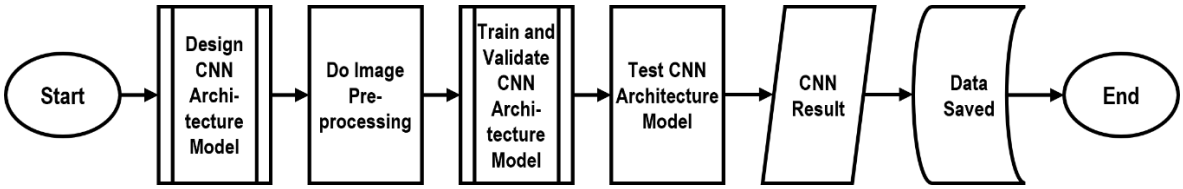


Figure 2. Classification system block

2.4. ResNet-9 architecture

Residual networks (ResNet) is a classic neural network which allows for deep training of neural networks [35]. ResNet 9 architecture, which has 9 layers, can be seen in Figure 3. ResNet-9 has a pretrained network that can classify images of up to 1,000 object categories which makes the network used to learn good feature representation for various images. ResNet 9 can be used as an image classification. ResNet 9 architecture has the concept of short connections, which will forward the input of each layer to the next layer. This concept will reduce the occurrence of errors or loss of features in the convolutional stage. ResNet 9 has 5 stages in the residual layer and is processed convolutionally and will be forwarded to the max pooling and fully connected layer.

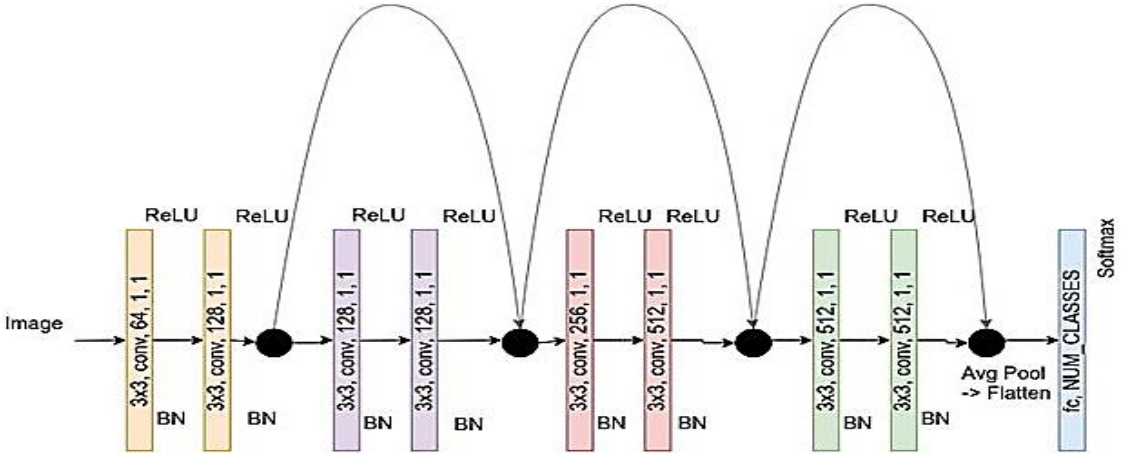


Figure 3. ResNet 9 architecture [36]

2.5. Adam optimizer

Adaptive moment estimation (Adam) is an algorithm for gradient descent optimization techniques. This method is very efficient when working with large problems involving a lot of data or parameters for which the algorithm is concerned. Adam has advantages over other optimizers, namely adaptive gradient, and root mean square propagation (RMSProp) [37], [38]. In determining the weight value if RMSProp uses the first value in the gradient, on the other hand, Adam uses the second value from the gradient. It is specifically able to calculate the calculated exponential mean of the gradient, the gradient squared, and the beta1/beta2 parameter controls the decaying average. The result of the comparison of Adam optimizer algorithm with other optimization algorithms shows that Adam optimization works well in practice [38]. The first way the Adam optimizer works is to measure the steps of the optimization to be made and determine the exponential decay rate for the forecast to be performed.

2.6. Designing a system overview

In this section, we will explain the general description of the system to be built, namely a system for classifying diseases in corn using CNN. This design is made as a reference for building a system on the hardware and software parts such as an overview of the system to be built, the design of the system, and the

output of the system. This design is expected to facilitate the implementation of the system. In the general description of the system, it can be seen that the components must be connected to each other in order to communicate to run the system.

Based on Figure 4, it can be seen that the system workflow on the web interface to be built is that users with various devices or software accessing the web interface to be built will make requests. The request that will be made is to FastAPI which is the API endpoint used in web interface development. In this FastAPI, requests from users, namely data in the form of images, will be redirected or will call the CNN model, namely the CNN model with ResNet-9 that has been formed previously to make predictions on the data that has been received. The result of the prediction will be sent back to the user via hypertext transfer protocol (HTTP) response, where the user will receive the predicted result in the form of a predicted or a label from the image and also confidence which is the accuracy value of the image prediction which will then be displayed on the web interface that is being run by the user.

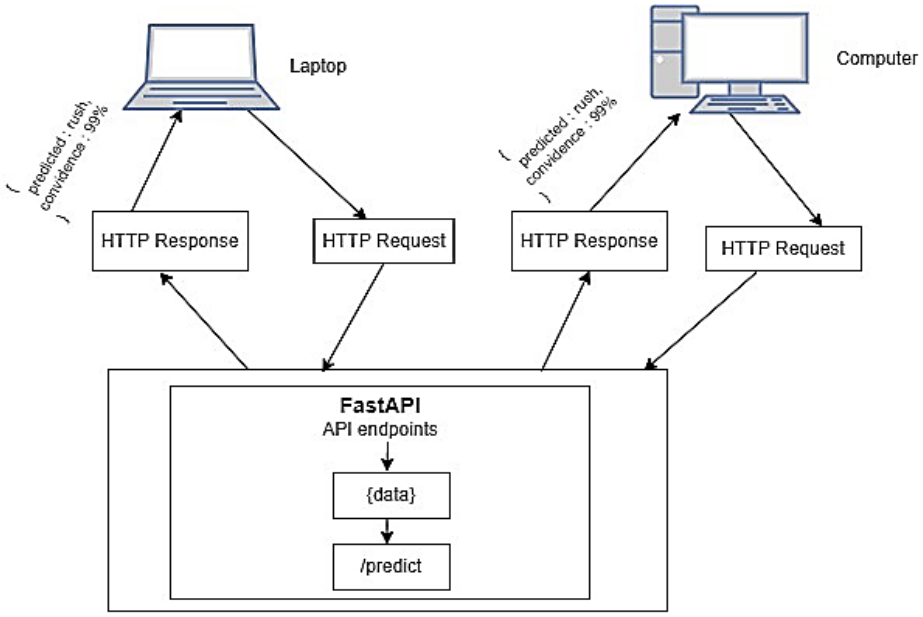


Figure 4. General description system

3. RESULT AND DISCUSSION

3.1. Result of hyperparameter

In deep learning, there are several hyperparameters that become variables that will affect model output, determine training details and significantly influence model performance [39]–[41]. In this study, three hyperparameter tuning experiments were carried out and can be shown in Table 1. Based on the table, *batch_size* is the number of sample data to be sent to the neural network. In this study, there are 9,165 datasets and the *batch_size* used is 32 sizes, which means using the first 32 data samples, it will be sent to the neural network, then the next 32 data samples until all data is distributed to the central network. The effect of the *batch_size* when running the training model is that the smaller the *batch_size* used, the smaller the memory used so it does not take a long time to run the training. *num_workers* is the number of how many sub-processes used to load data. If the default $number = 0$ means the data will be loaded in the main process, in other words, the main process will do the data loading when needed.

Table 1. Hyperparameter tunning experiment

Experiment Number	Hyperparameter Tunning Experiment	Run Time for Model Evaluation	Run Time for Model Training
1	batch_size: 64	30 minutes	5 minutes
	num_workers: 2	5 seconds	24 seconds
2	batch_size: 64	6 minutes	5 minutes
	num_workers: 4	36 seconds	24 seconds
3	batch_size: 32	31 seconds	4 minutes
	num_workers: 4		50 seconds

Based on the results of hyperparameter tuning experiments carried out 3 times in Table 1, it can be concluded that if the *num_workers* used is larger and the *batch_size* per iteration is smaller it will affect the runtime speed where the required runtime will be less or faster in running the model. So, this study applies the results of the 3 hyperparameter tuning experiments where *num_workers* is 4 and *batch_size* is 32. This hyperparameter tuning experiment is carried out in the data preparation process by changing the size of *batch_size*, evaluating the model, and also training the model by changing the value set for the number of *num_workers*.

3.2. Result of building model

ResNet-9 model building process which shows that at each convolution layer a batch normalization layer is added to normalize the output of the previous layer, then followed by an activation layer to neutralize values less than 0 and max pooling will be carried out to obtain the results maximum of all image pixels. Figure 2 shows the model-building process and summary of the output of the ResNet-9 model which uses the torch summary library. Figure 5 describes the implementation of the CNN architecture for the formation of the model used for training. Basically, first, we resize each image to 256×256. After that, the image is inserted into the CNN. The first convolution layer applies 32 filter sizes or channels output. That means 32 different filters are applied to the image and try to find the features and after that using 32 features the author creates a feature map that has 32 channels. So, from 3×256×256 to 32×254×254. After that, the ReLU activation function will be applied to eliminate non-linearity.

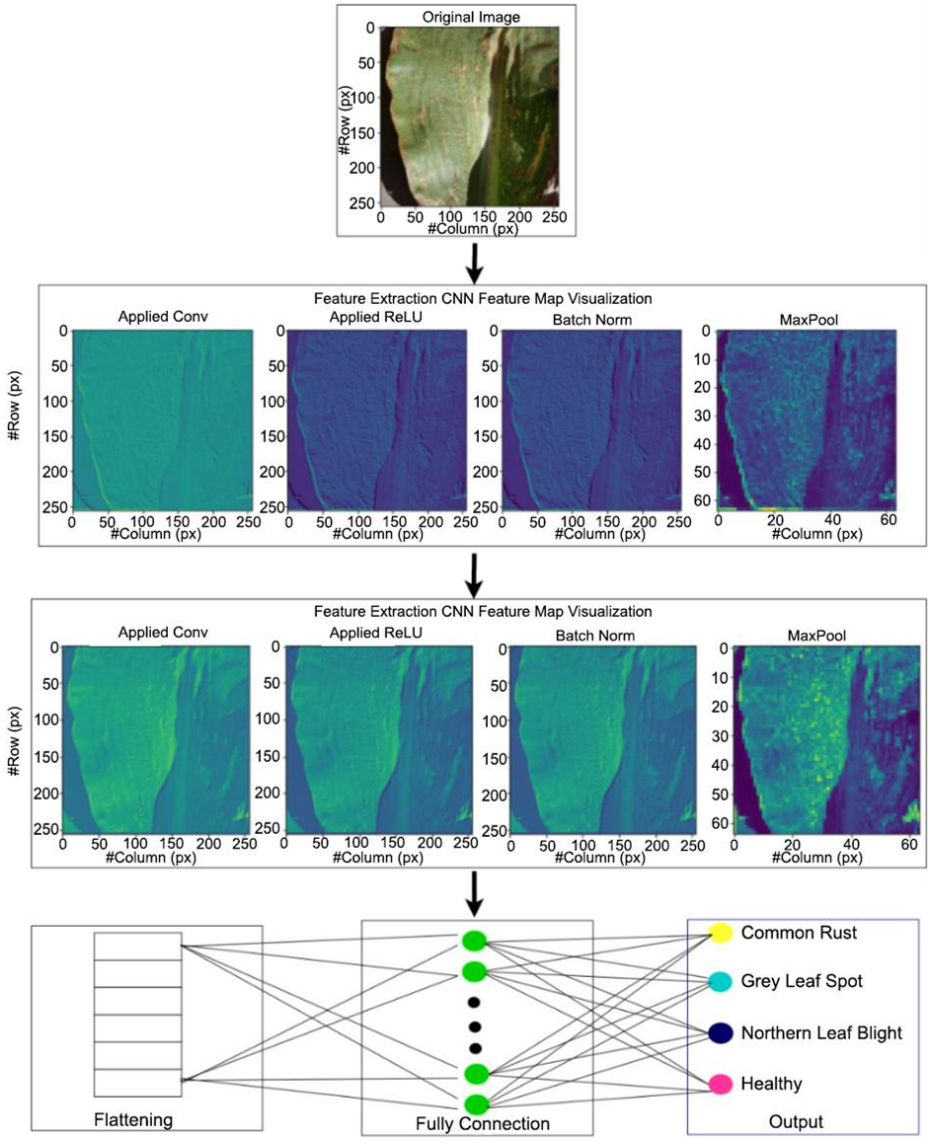


Figure 5. Model building process

Batch normalization to normalize neuron weights. After that, we put this image into the maximum batch layer taking only the most relevant features, so we get an output image in the form of 32×128×128. After that, we put this image into the next convolution layer and the process is the same as mentioned above. Then we flatten the output of the final maximum pooling layer and insert it into the next linear layer, which is also called the fully connected layer, and finally, as the last layer, and predict 3 categories. So as the output model, we get a tensor size of 1×3. From the tensor, the index is taken from the maximum value of the tensor. That particular index is our main prediction. The visual display of the image of the corn plant while going through the training and classification process on the CNN method as shown in Figure 6.

From Table 2, it can be seen that this study uses 9 layers in the construction of the model because the model used is ResNet-9 architecture, where each layer has a combination of convolution, normalization, ReLU, and max pooling, which is used in forming the model, where the convolution modeling is done eight times and then proceed to flatten. The summary will be based on the input shape used in this study, namely 3×256×256 which explains that the image used has 256×256 pixels with 3 RGB channels. For the explanation of the first output shape, the applied filter is 3×3 (kernel size) and will move with a stride of 1×1, for a pixel size of 256×256 this will produce 64 filters. The first convolution on the first layer will bring 3×3×3 (according to the output dimensions) as big as 27 params + 1 bios in total is 28 filters for each move. The total parameters for the first convolution generated are 28×64 filters of 1,792. Then normalization will be carried out using Batchnorm2D so that the total params carried are 2×64 (filter size) of 128. For activation, it still maintains its previous form, so it requires 0 params for the process of maintaining it.

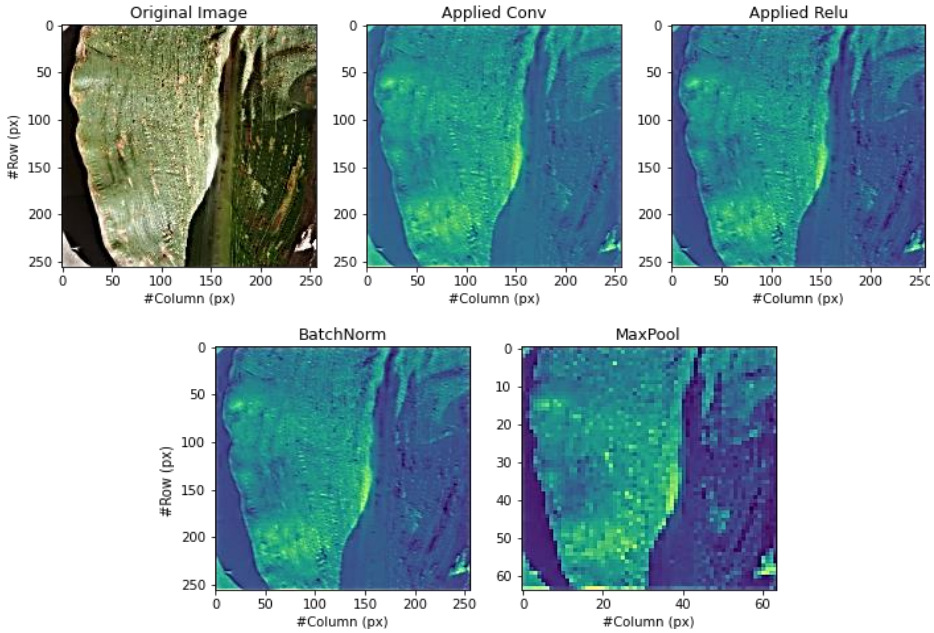


Figure 6. Feature extraction CNN feature map visualization

Table 2. Summary ResNet-9 model

Layer (type)	Output Shape	Params	Layer (type)	Output Shape	Params
Conv2d-1	[−1, 64, 256, 256]	1792	ReLU-16	[−1, 256, 64, 64]	0
BatchNorm2d-2	[−1, 64, 256, 256]	128	MaxPool2d-17	[−1, 256, 16, 16]	0
ReLU-3	[−1, 64, 256, 256]	0	Conv2d-18	[−1, 512, 16, 16]	1180160
Conv2d-4	[−1, 128, 256, 256]	73856	BatchNorm2d-19	[−1, 512, 16, 16]	1024
BatchNorm2d-5	[−1, 128, 256, 256]	256	ReLU-20	[−1, 512, 16, 16]	0
ReLU-6	[−1, 128, 256, 256]	0	MaxPool2d-21	[−1, 512, 4, 4]	0
MaxPool2d-7	[−1, 128, 64, 64]	0	Conv2d-22	[−1, 512, 4, 4]	2359808
Conv2d-8	[−1, 128, 64, 64]	147584	BatchNorm2d-23	[−1, 512, 4, 4]	1024
BatchNorm2d-9	[−1, 128, 64, 64]	256	ReLU-24	[−1, 512, 4, 4]	0
ReLU-10	[−1, 128, 64, 64]	0	Conv2d-25	[−1, 512, 4, 4]	2359808
Conv2d-11	[−1, 128, 64, 64]	147584	BatchNorm2d-26	[−1, 512, 4, 4]	1024
BatchNorm2d-12	[−1, 128, 64, 64]	256	ReLU-27	[−1, 512, 4, 4]	0
ReLU-13	[−1, 128, 64, 64]	0	MaxPool2d-28	[−1, 512, 1, 1]	0
Conv2d-14	[−1, 256, 64, 64]	295168	Flatten-29	[−1, 512]	0
BatchNorm2d-15	[−1, 256, 64, 64]	512	Linear-30	[−1, 4]	2052

3.3. Result of training model

After implementing the model formation, then the model that has been formed will be trained using Adam optimizer. Table 3 is the result of the training model with 100 epoch experiments. Training loss is the value of calculating the loss function of the training dataset and predictions from the model. Validation loss is the calculation value of the loss function from the validation dataset and predictions from the model with input data from the validation dataset. Validation accuracy is the value of calculating the accuracy of the validation dataset and predictions from the model with input data from the validation dataset [42]. From the table, it can be concluded that the higher the number of epochs, the higher the value of validation accuracy and the lower the value of validation loss. The final validation accuracy value obtained after running 100 epochs is 0.9908 and the running time required to run 100 epochs is 1 hour 41 minutes 2 seconds.

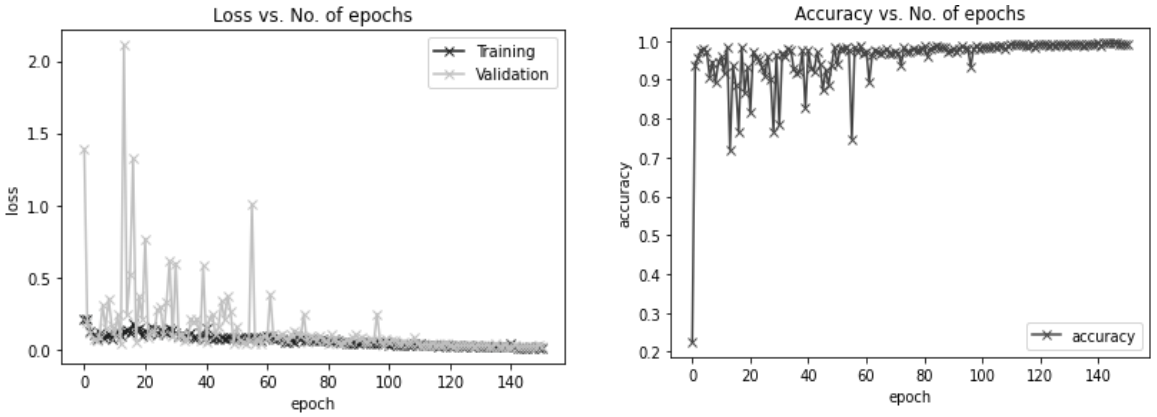
Table 3. Experiment result 100 epoch

Epoch	Training Loss	Validation Loss	Validation Accuracy	Epoch	Training Loss	Validation Loss	Validation Accuracy
1	0.2193	0.0306	0.8464	51	0.0364	0.0923	0.9714
2	0.1987	0.0300	0.4240	52	0.0434	0.0906	0.9741
3	0.2129	0.0296	0.9302	53	0.0434	0.0430	0.9855
4	0.2305	0.0529	0.8141	54	0.0332	0.0332	0.9908
5	0.2458	1.0056	0.7836	55	0.0326	0.0450	0.9887
6	0.2642	0.2954	0.8997	56	0.0333	0.0506	0.9914
7	0.2268	3.0146	0.5399	57	0.0374	0.0358	0.9828
8	0.1465	0.1531	0.9434	58	0.0311	0.0442	0.9849
9	0.1473	0.1021	0.9553	59	0.0317	0.0400	0.9898
10	0.1574	3.8027	0.4319	60	0.0292	0.0355	0.9892
11	0.1919	0.1411	0.9456	61	0.0250	0.0458	0.9887
12	0.1442	0.3598	0.8941	62	0.0271	0.0493	0.9871
…	…	…	…	…	…	…	…
46	0.0311	0.0281	0.9898	96	0.0191	0.0275	0.9903
47	0.0484	0.0377	0.9887	97	0.0176	0.0237	0.9919
48	0.0304	0.0280	0.9898	98	0.0163	0.0258	0.9919
49	0.0556	0.0315	0.9881	99	0.0161	0.0262	0.9914
50	0.0325	0.0294	0.9903	100	0.0150	0.0250	0.9908

Table 4 shows experimental data from the training dataset with several experiments for the number of epochs, this is done in order to find a high level of accuracy for the model formed. The conclusion that can be drawn is that more ages will be used for a longer time but will increase the accuracy of the model formed. From Figure 7(a), it can be seen that the graph displays the loss value obtained from each epoch with a total of 100 epochs, where the loss value obtained in the training data is marked with a blue line graph and for validation data, it is marked in red. In the training data, the loss value in the first epoch is 0.0213, in the second epoch it is 0.0300, in the third epoch it is 0.0186, in the fourth epoch 0.0325, in the fifth epoch it is 0.0203 until the 100th epoch is 0.0150. In validation data, the loss value obtained in the first epoch is 0.0296, at the second epoch is 4.643, at the third epoch is 0.0296, at the fourth epoch is 0.0529, in the fifth epoch is 0.0262 until the 100th epoch the loss value is 0.0373, where the more the number of epochs, the lower the loss value obtained and if the graph displayed is optimal, it can be dismissed. Based on the picture, it can also be seen that the optimal graph is at 100 epochs, so for a total of 100 epochs, the loss value is considered optimal. From Figure 7(b), it can be seen that the graph displays the accuracy value obtained from each epoch with a total of 100 epochs, where the accuracy obtained in the first epoch is 0.635, the second epoch is 0.837, the third epoch is 0.954, the fourth epoch is 0.973, the fifth epoch is 0.985 and so on until the 100th epoch value is 0.9908, where based on the epoch experiment that has been done, it can be concluded that the more the number of epochs, the higher the accuracy value obtained.

Table 4. Dataset training results

Experiment Number	Data Dividing	Epoch	Run Time (Minute)	Accuracy
1	Training data 80% and validation data 20%	5 epoch	17	0.9806
2	Training data 80% and validation data 20%	25 epoch	51	0.9892
3	Training data 80% and validation data 20%	55 epoch	66	0.9898
4	Training data 80% and validation data 20%	75 epoch	72	0.9903
5	Training data 80% and validation data 20%	100 epoch	101	0.9908



(a) (b)

Figure 7. Epoch trial graph of 100 times for (a) loss value (b) accuracy value

3.4. Result of web interface

In this section, we will explain n the results of the web interface that has been successfully built to make it easier for users to use the CNN model in classifying diseases in corn plants. Figure 8(a) is the web interface of the input image that has been built which will display an image drop box that can be used to upload images. After uploading the image, then the image predicting process that has been built will display the predicting process with the display of images that have been successfully uploaded or inputted previously and also the "Processing" status. Figure 8(b) is a web interface of the output or result predicting image that has been built which will display the accuracy results and also the label of the predicted image that has been uploaded or inputted previously.

(a) (b)

Figure 8. Web interface corn plant classification (a) input image and (b) result image

3.5. Result of testing

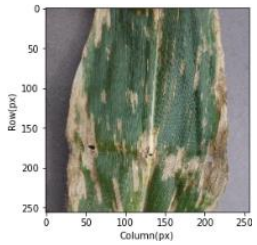
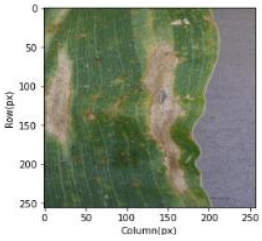
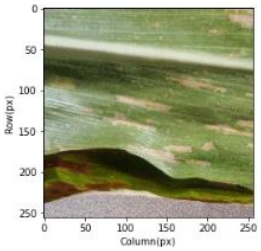
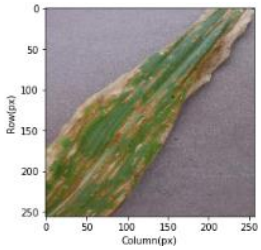
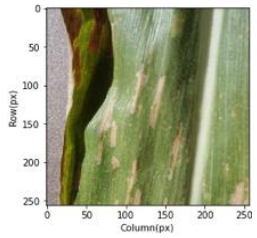
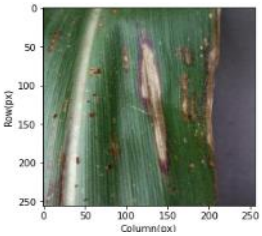
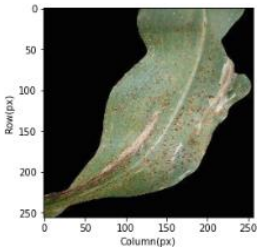
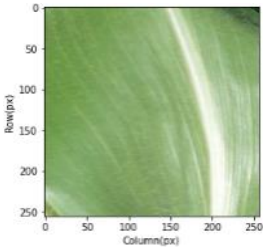
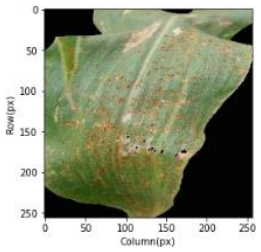
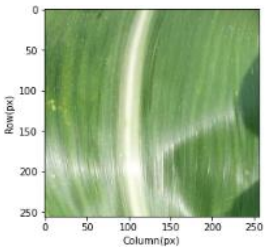
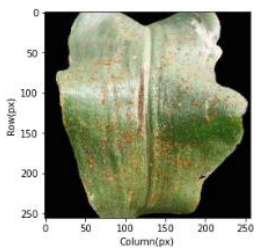
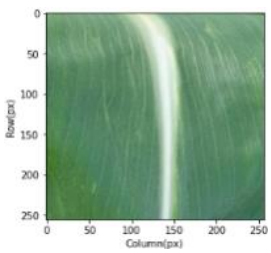
This stage will explain the test results from the data that has been tested using CNN. Testing in this study was carried out using a model that had been trained or trained previously. The test data from this research will be tested using the training model that has been built. The test result data which are gray leaf spot, common rust, northern leaf blight, and healthy consisting of image results, detection results, and accuracy test results can be seen in Table 5.

3.6. Result of evaluation

In this section, the performance metrics from the confusion matrix will be explained which are used to calculate various performance metrics in measuring the performance of the model that has been created. In working on the confusion matrix in this study, the amount of data used is 9165 data obtained from training data and valid data. Training data as much as 1,916 for common rust, 1,651 for *Cercospora* leaf spot gray

leaf spot, 1,907 for northern leaf blight, and for healthy there are 1,858 images. as for valid data, 478 were taken for common rust, 411 for *Cercospora* leaf spot gray leaf spot, 478 for northern leaf blight, and for healthy there were 466 images, where the total data is 9,165 data, and the data for each category is 2,394 for common rust, 2,062 for gray leaf spot, 2,385 for northern leaf blight, and 2,324 for healthy. The confusion matrix result can be shown in Table 6.

Table 5. Test result testing of gray leaf spot, common rust, northern leaf blight and healthy

Picture	Result	Accuracy	Picture	Result	Accuracy
	Detected Gray leaf spot	99.99		Detected Northern Leaf Blight	99.99
	Detected Gray leaf spot	99.94		Detected Northern Leaf Blight	99.46
	Detected Gray leaf spot	97.81		Detected Northern Leaf Blight	99.15
	Detected Common Rust	100.0		Detected Healthy	99.99
	Detected Common Rust	99.94		Detected Healthy	99.98
	Detected Common Rust	100.0		Detected Healthy	99.98

Based on the data from Table 5 obtained values for true positive (TP), false negative (FN), false positive (FP), and also true negative (TN) from each category, namely gray leaf spot (GLS), healthy, northern leaf blight (NLB), and rust, where this value will be used to calculate the value of the performance metrics in this study, namely using the accuracy value. Accuracy is what describes how accurate the model is in classifying correctly or is the level of closeness of the predicted value to the actual value [26]. The accuracy value obtained based on calculations using the accuracy formula is 0.99, which means that the accuracy value for correctly predicting data from the entire dataset is 99%.

$$Accuracy = \frac{TP}{Total\ Dataset} = \frac{9128}{9165} = 0.995 \quad (1)$$

Table 6. Confusion matrix result

Actual	Prediction			
	GLS	NLB	Rust	Healthy
GLS	2046	0	16	0
NLB	0	2324	0	0
Rust	15	3	2366	1
Healthy	0	0	2	2392

4. CONCLUSION

The amount of data used in this study affects the quality of the resulting model. The dataset used in this study is 9,165 image data with 4 categories in building a CNN model, which in this study also compares 5, 25, 55, 75, and 100 epochs for the model to be used. The highest accuracy value in the experiment was obtained from experiments using 100 epochs so in this study 100 epochs were used in model formation. Based on the results of the hyperparameter tuning experiments carried out, it can be concluded that if the *num_workers* used is larger and the *batch_size* per iteration is smaller it will affect the runtime speed where the required runtime will be less or faster in running the model. Therefore, this study applies the results of hyperparameter tuning experiments where *num_workers* is 4 and *batch_size* is 32. The model produced in this study was able to detect the type of disease in corn plants through leaf imagery with good accuracy where the results obtained were 99%. In this study, a simple web interface has been developed for classifying corn leaf images where the user system will enter an image, then the system will predict the image and display the classification results from the image that has been inputted.

REFERENCES

[1] H. Sinay and N. Harijati, "Determination of proximate composition of local corn cultivar from Kisar Island, Southwest Maluku Regency," *Biosaintifika: Journal of Biology & Biology Education*, vol. 13, no. 3, pp. 258–266, Dec. 2021, doi: 10.15294/biosaintifika.v13i3.30527.

[2] K. Astuti, O. R. Prasetyo, and I. N. Khasanah, "The 2020 analysis of maize and soybean productivity in Indonesia (The results of crop cutting survey)," BPS-Statistics Indonesia, 2020.

[3] A. Sofowora, E. Ogunbodede, and A. Onayade, "The role and place of medicinal plants in the strategies for disease prevention," *African Journal of Traditional, Complementary and Alternative Medicines*, vol. 10, no. 5, pp. 210–229, Aug. 2013, doi: 10.4314/ajtcam.v10i5.2.

[4] V. Gorshkov and I. Tsers, "Plant susceptible responses: the underestimated side of plant–pathogen interactions," *Biological Reviews*, vol. 97, no. 1, pp. 45–66, Feb. 2022, doi: 10.1111/brv.12789.

[5] A. Hidayat, U. Darusalam, and I. Irmawati, "Detection of disease on corn plants using convolutional neural network methods," *Jurnal Ilmu Komputer dan Informasi*, vol. 12, no. 1, pp. 51–56, Mar. 2019, doi: 10.21609/jiki.v12i1.695.

[6] R. Hu, S. Zhang, P. Wang, G. Xu, D. Wang, and Y. Qian, "The identification of corn leaf diseases based on transfer learning and data augmentation," in *Proceedings of the 2020 3rd International Conference on Computer Science and Software Engineering*, May 2020, pp. 58–65, doi: 10.1145/3403746.3403905.

[7] J. Chen, W. Wang, D. Zhang, A. Zeb, and Y. A. Nanehkaran, "Attention embedded lightweight network for maize disease recognition," *Plant Pathology*, vol. 70, no. 3, pp. 630–642, Apr. 2021, doi: 10.1111/ppa.13322.

[8] X. Qian, C. Zhang, L. Chen, and K. Li, "Deep learning-based identification of maize leaf diseases is improved by an attention mechanism: Self-attention," *Frontiers in Plant Science*, vol. 13, Apr. 2022, doi: 10.3389/fpls.2022.864486.

[9] S. Patil, P. Oswal, V. Sukenkar, and S. Choudhary, "Corn leaf disease detection-a comparative study using transfer learning techniques," *International Research Journal of Modernization in Engineering Technology and Science*, vol. 4, no. 5, pp. 5752–5756, 2022.

[10] J. Sucher *et al.*, "The durable wheat disease resistance gene Lr34 confers common rust and northern corn leaf blight resistance in maize," *Plant Biotechnology Journal*, vol. 15, no. 4, pp. 489–496, Apr. 2017, doi: 10.1111/pbi.12647.

[11] H. A. Craze, N. Pillay, F. Joubert, and D. K. Berger, "Deep learning diagnostics of gray leaf spot in maize under mixed disease field conditions," *Plants*, vol. 11, no. 15, Jul. 2022, doi: 10.3390/plants11151942.

[12] N. R. Abdelsalam *et al.*, "Inheritance of resistance against northern leaf blight of maize using conventional breeding methods," *Saudi Journal of Biological Sciences*, vol. 29, no. 3, pp. 1747–1759, Mar. 2022, doi: 10.1016/j.sjbs.2021.10.055.

[13] R. Vinuesa *et al.*, "The role of artificial intelligence in achieving the sustainable development goals," *Nature Communications*, vol. 11, no. 1, Dec. 2020, doi: 10.1038/s41467-019-14108-y.

[14] I. H. Sarker, "Machine learning: Algorithms, real-world applications and research directions," *SN Computer Science*, vol. 2, no. 3, May 2021, doi: 10.1007/s42979-021-00592-x.
[15] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," *Frontiers in Plant Science*, vol. 7, Sep. 2016, doi: 10.3389/fpls.2016.01419.
[16] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, Nov. 2018, doi: 10.1016/j.heliyon.2018.e00938.
[17] J. Lu, L. Tan, and H. Jiang, "Review on convolutional neural network (CNN) applied to plant leaf disease classification," *Agriculture*, vol. 11, no. 8, Jul. 2021, doi: 10.3390/agriculture11080707.
[18] K. R. Aravind, P. Raja, K. V Mukesh, R. Aniirudh, R. Ashiwin, and C. Szczepanski, "Disease classification in maize crop using bag of features and multiclass support vector machine," in *2018 2nd International Conference on Inventive Systems and Control (ICISC)*, Jan. 2018, pp. 1191–1196, doi: 10.1109/ICISC.2018.8398993.
[19] S. S. Athani and C. Tejeshwar, "Support vector machine-based classification scheme of maize crop," in *2017 IEEE 7th International Advance Computing Conference (IACC)*, Jan. 2017, pp. 84–88, doi: 10.1109/IACC.2017.0032.
[20] B. K. Khotimah, "Performance of the K-nearest neighbors method on identification of maize plant nutrients," *Jurnal Infotel*, vol. 14, no. 1, pp. 8–14, Feb. 2022, doi: 10.20895/infotel.v14i1.735.
[21] P. Tiwari *et al.*, "CNN based multiclass brain tumor detection using medical imaging," *Computational Intelligence and Neuroscience*, vol. 2022, pp. 1–8, Jun. 2022, doi: 10.1155/2022/1830010.
[22] S. Sony, K. Dunphy, A. Sadhu, and M. Capretz, "A systematic review of convolutional neural network-based structural condition assessment techniques," *Engineering Structures*, vol. 226, Jan. 2021, doi: 10.1016/j.engstruct.2020.111347.
[23] J. Gupta, S. Pathak, and G. Kumar, "Deep learning (CNN) and transfer learning: A review," *Journal of Physics: Conference Series*, vol. 2273, no. 1, May 2022, doi: 10.1088/1742-6596/2273/1/012029.
[24] L. Alzubaidi *et al.*, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *Journal of Big Data*, vol. 8, no. 1, Dec. 2021, doi: 10.1186/s40537-021-00444-8.
[25] X. Sun, G. Li, P. Qu, X. Xie, X. Pan, and W. Zhang, "Research on plant disease identification based on CNN," *Cognitive Robotics*, vol. 2, pp. 155–163, 2022, doi: 10.1016/j.cogr.2022.07.001.
[26] T.-M. Le, N.-P.-L. Le, N.-G.-N. Hua, N.-L. Bui, and V. Q. Dinh, "Insight evaluation on traditional and CNN features," in *2022 The 6th International Conference on Machine Learning and Soft Computing*, Jan. 2022, pp. 179–185, doi: 10.1145/3523150.3523178.
[27] S. Izadi, M. Ahmadi, and R. Nikbazm, "Network traffic classification using convolutional neural network and ant-lion optimization," *Computers and Electrical Engineering*, vol. 101, Jul. 2022, doi: 10.1016/j.compeleceng.2022.108024.
[28] M. S. Hanif and M. Bilal, "Competitive residual neural network for image classification," *ICT Express*, vol. 6, no. 1, pp. 28–37, Mar. 2020, doi: 10.1016/j.icte.2019.06.001.
[29] K. E. Tokarev, V. M. Zotov, V. N. Khavronina, and O. V Rodionova, "Convolutional neural network of deep learning in computer vision and image classification problems," *IOP Conference Series: Earth and Environmental Science*, vol. 786, no. 1, Jun. 2021, doi: 10.1088/1755-1315/786/1/012040.
[30] H. Ismail, A. F. M. Ayob, A. M. S. M. Muslim, and M. F. R. Zulkifli, "Convolutional neural network architectures performance evaluation for fish species classification," *Journal of Sustainability Science and Management*, vol. 16, no. 5, pp. 124–139, Jul. 2021, doi: 10.46754/jssm.2021.07.010.
[31] M. Manav, M. Goyal, A. Kumar, A. K. Arya, H. Singh, and A. K. Yadav, "Deep learning approach for analyzing the COVID-19 chest X-rays," *Journal of Medical Physics*, vol. 46, no. 3, pp. 189–196, 2021, doi: 10.4103/jmp.JMP_22_21.
[32] J. Kaur and W. Singh, "Tools, techniques, datasets and application areas for object detection in an image: a review," *Multimedia Tools and Applications*, vol. 81, no. 27, pp. 38297–38351, Nov. 2022, doi: 10.1007/s11042-022-13153-y.
[33] H.-C. Chen *et al.*, "AlexNet convolutional neural network for disease detection and classification of tomato leaf," *Electronics*, vol. 11, no. 6, Mar. 2022, doi: 10.3390/electronics11060951.
[34] C. Fan, M. Chen, X. Wang, J. Wang, and B. Huang, "A review on data preprocessing techniques toward efficient and reliable knowledge discovery from building operational data," *Frontiers in Energy Research*, vol. 9, Mar. 2021, doi: 10.3389/fenrg.2021.652801.
[35] H. Wang, K. Li, and C. Xu, "A new generation of ResNet model based on artificial intelligence and few data driven and its construction in image recognition model," *Computational Intelligence and Neuroscience*, vol. 2022, pp. 1–10, Mar. 2022, doi: 10.1155/2022/5976155.
[36] C. Rajagopalan, D. Rawlinson, E. Goldberg, and G. Kowadlo, "Deep learning in a bilateral brain with hemispheric specialization," *arXiv:2209.06862*, 2022.
[37] A. F. ud Din *et al.*, "Deep reinforcement learning for integrated non-linear control of autonomous UAVs," *Processes*, vol. 10, no. 7, Jul. 2022, doi: 10.3390/pr10071307.
[38] Y. Wang, Z. Xiao, and G. Cao, "A convolutional neural network method based on Adam optimizer with power-exponential learning rate for bearing fault diagnosis," *Journal of Vibroengineering*, vol. 24, no. 4, pp. 666–678, Jun. 2022, doi: 10.21595/jve.2022.22271.
[39] J. Kalliola, J. Kapočiūtė-Dzikienė, and R. Damaševičius, "Neural network hyperparameter optimization for prediction of real estate prices in Helsinki," *PeerJ Computer Science*, vol. 7, Apr. 2021, doi: 10.7717/peerj-cs.444.
[40] I. H. Sarker, "Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions," *SN Computer Science*, vol. 2, no. 6, Nov. 2021, doi: 10.1007/s42979-021-00815-1.
[41] D. Passos and P. Mishra, "A tutorial on automatic hyperparameter tuning of deep spectral modelling for regression and classification tasks," *Chemometrics and Intelligent Laboratory Systems*, vol. 223, Apr. 2022, doi: 10.1016/j.chemolab.2022.104520.
[42] M. Bilal and A. A. Almazroi, "Effectiveness of fine-tuned BERT model in classification of helpful and unhelpful online customer reviews," *Electronic Commerce Research*, Apr. 2022, doi: 10.1007/s10660-022-09560-w.

BIOGRAPHIES OF AUTHORS

Tegar Arifin Prasetyo is currently a senior lecturer and research member in Information Technology Department at Institut Teknologi Del since 2020. He has experience specializing in building mathematical models, machine learning, analytical android tools development, control system, and computer programming. He dedicates himself to university teaching and conducting research. His research interests include artificial intelligence, machine learning, computational algorithm, optimal control, a mathematical model in epidemiology, and bioinformatics. He can be contacted at tegar.prasetyo@del.ac.id or arifintegar12@gmail.com.

Victor Lambok Desrony is now a third-year student in Information Technology Department at Institut Teknologi Del. He has experience in classification diseases using neural networks, back-end/infrastructure web and desktop development, and microservice system. His research interests include artificial intelligence and machine learning. He can be contacted at victordesrony@gmail.com.

Henny Flora Panjaitan is now a third-year student in Information Technology Department at Institut Teknologi Del. She has experience in classification diseases using neural networks, web and desktop development, and UX design. Her research interests include machine learning algorithms. She can be contacted at hennypanjaitan5@gmail.com.

Romauli Sianipar is now a third-year student in Information Technology Department at Institut Teknologi Del. She has experience in classification diseases using neural networks, Android and web developing, and computer programming. Her research interests include image processing using neural networks and computational programming. She can be contacted at romaulimauliate@gmail.com.

Yohanssen Pratama is currently a senior lecturer and researcher member in Software Engineering Technology Department at Institut Teknologi Del. He has more than five-year experience specializing in back-end/infrastructure, analytical tools development and computer programming. Teach academic and vocational subjects to undergraduate also pursue my own research to contribute to the wider research activities of my department. He can be contacted at yohanssen.pratama@del.ac.id.