**Building of Informatics, Technology and Science (BITS)**
Volume 6, No 3, Desember 2024 Page: 1293−1300
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v6i3.5883

# Classification of Rice Plant Disease Image Using Convolutional Neural Network (CNN) Algorithm based on Amazon Web Service (AWS)

**Nova Anggraini, Bagus Adhi Kusuma, Pungkas Subarkah*, Fandy Setyo Utomo, Nandang Hermanto**

Informatics Study Program, Faculty of Computer Science, Universitas Amikom Purwokerto, Indonesia
Email:[1]novaanggraini213@gmail.com, [2]bagus@amikompurwokerto.ac.id, [3,*]subarkah@amikompurwokerto.ac.id,
[4]fandysetyoutomo@amikompurwokerto.ac.id, [5]nandanghermanto@amikompurwokerto.ac.id
Correspondence Author Email: subarkah@amikompurwokerto.ac.id
Submitted: **04/09/2024**; Accepted: **01/12/2024**; Published**: 03/12/2024**

**Abstract**−In agriculture, rice plays an important role in the Indonesian economy. Rice produces rice, one of the most widely consumed staple food sources in Indonesia. Many factors can cause rice production failure, one of which is leaf pests and diseases. Therefore, early identification and management of plant diseases is an important step in an effort to increase crop yields and ensure food safety. One way to detect rice leaf images early is to perform an image classification process and create a web-based application. The method that has the ability in image processing is deep learning technique with convolutional neural network (CNN) method. The Convolutional Neural Network (CNN) method works to perform and predict diseases in plants by using image categorization or object images. This research aims to apply the web application of image classification of rice plant diseases to the Amazon Web Service (AWS) by identifying and classifying various types of rice leaf diseases using the CNN algorithm, so that farmers can detect rice plant diseases quickly and accurately through image analysis. This application was created using Convolutional Neural Network (CNN) methodology and Software Development Life Cycle (SDLC). The result of this study is that researchers created a web application for the classification of rice plant diseases through leaf images which are divided into 4 categories, namely Healthy, Leaf Blight, Brown Leaf Blight and Hispa, which is made a classification model using CNN with an accuracy value of 0. 8608, then using the streamlit framework to build a website, and utilizing AWS services in the form of Amazon Elastic Compute Cloud (Amazon EC2) as a hosting service, Amazon Simple Storage Service (Amazon S3) as a service for storing rice plant disease classification models and for storing web files, and Amazon Identity and Access Management Role (Amazon IAM) as a service to create a role that gives permission to connect between AWS S3 and AWS EC2. Testing the disease classification model in rice plants implemented on the web in EC2 shows quite good results with an accuracy of 78.5%. This can affect the model's ability to recognize specific disease patterns.

**Keywords**: Rice leaf diseases; CNN; SDLC; Web Streamlit; Amazon Web Service

## 1. INTRODUCTION

In agriculture, rice plays an important role in the Indonesian economy. Rice produces rice, one of the most widely consumed staple food sources in Indonesia. The success rate of rice harvests is crucial as failure in rice production can result in economic and political instability [1]. Therefore, product quality and rice plant health are the most important aspects to maintain rice productivity. Many factors can cause the failure of rice crop production, one of which is leaf pests and diseases. However, rice cultivation is inseparable from various challenges, one of which is the attack of plant diseases. Rice plant diseases such as blast, and brown leaf spot can cause significant losses, even reaching 20-50% of total production if not handled properly [2]. Therefore, early identification and management of plant diseases is an important step in efforts to increase crop yields and ensure food safety. However, the manual identification process of rice diseases has several limitations. First, identification by farmers or agricultural experts often takes a long time and requires specialized skills. Second, manual identification tends to be subjective and prone to errors, especially if disease symptoms resemble each other. Thirdly, in remote rural areas, access to agricultural experts capable of accurately diagnosing diseases can be very limited. Therefore, a rice plant disease detector is needed to minimize losses.

With the development of technology, one of the alternatives that can also be done to overcome existing limitations is to provide an application that can help farmers in detecting early various types of diseases that attack rice plants. The application can be used by all farmers anywhere and anytime [3]. Along with technological developments, especially in the fields of artificial intelligence (AI) and machine learning (ML), opportunities arise to overcome these challenges through innovative technological solutions [4]. AI and ML technologies can be used to develop an image classification system that is able to recognize and identify rice plant diseases with high accuracy. The system can be trained using a dataset of images of rice plant diseases that have been labeled, so that it is able to detect disease symptoms based on the patterns and characteristics present in the images [5].

Therefore, one way to detect rice leaf images early is by processing image classification and making web-based applications. The method that has the ability in image processing is deep learning technique with convolutional neural network (CNN) method [6][7]. The Convolutional Neural Network (CNN) method works to conduct and predict diseases in plants by using image categorization or object images [8] [9]. In addition, after creating a classification model, researchers implemented it into a web application using the streamlit framework. Streamlit is an open source framework with Python programming language that facilitates the development of web applications especially in data science and machine learning, as it has several features that make it easy to develop machine learning model. Streamlit provides visualization capabilities through an interactive interface to communicate relevant information to the user. The implementation of Streamlit is quite simple through the Streamlit cloud sharing platform and can be accessed

Copyright © 2024 **Author**, Page 1293
This Journal is licensed under a Creative Commons Attribution 4.0 International License

**Building of Informatics, Technology and Science (BITS)**
Volume 6, No 3, Desember 2024 Page: 1293−1300
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v6i3.5883

across multiple platforms [10]. The streamlit application that has been implemented through CNN will also be implemented through Amazon Web Service (AWS) to store hosting, storing data and maintaining the security of rice plant disease data. Amazon Web Services (AWS) is a cloud computing platform offered by Amazon. AWS provides a comprehensive range of cloud services, including computing, storage, databases, analytics, networking, security, and more, allowing individuals and businesses to create and run applications and services with great scalability, dependability, and cost efficiency. AWS is one of the world's largest and most popular cloud service providers, used by small and large businesses for a variety of information technology needs [11].

In this study, researchers used Amazon Web Service (AWS) capabilities such as Amazon Elastic Compute Cloud (Amazon EC2), Amazon Simple Storage Service (Amazon S3), and AWS IAM (Identity and Access Management) to create efficient web applications [12]. Amazon Elastic Compute Cloud (Amazon EC2) is a web service that offers safe, scalable computing capability in the cloud [13]. With Amazon EC2, there are many options that help build and run almost any application [14]. Amazon Simple object storage service (Amazon S3) offers the best performance, scalability, security and data availability on the market. For almost any use case, customers of any size and industry can store and protect a lot of data, such as data lakes, cloud-native applications, and mobile applications. Easy-to-use management features and cost-effective storage classes allow you to optimize costs, organize data, and customize access controls to meet specific needs, organizations, and compliance [15]. Meanwhile, AWS Identity and Access Management (IAM) Roles are entities that you create and assign specific permissions that allow trusted identities, such as workforce and application identities, to perform actions in AWS. When your trusted identities take on an IAM Role, they will only be granted permissions covered by the IAM Role [16].

Based on the above background, researchers use the CNN method in this study to classify the type of rice disease through leaf images which are divided into 4 categories, namely Healthy, LeafBlast, BrownSpot and Hispa [17]. In addition, in making this web-based application using stremlite to build a rice plant detection website which later researchers will utilize AWS services in the form of Amazon Elastic Compute Cloud (Amazon EC2) as a hosting service, Amazon Simple Storage Service (Amazon S3) as a service to store a rice plant disease classification model and to store web files that use the streamlit framework, and Amazon Identity and Access Management Role (Amazon IAM) as a service to create a role that gives permission to connect between AWS S3 and AWS EC2.

The purpose of this research is to implement a rice plant disease image classification web application using the streamlit framework to the Amazon Web Service (AWS) service by identifying and classifying the type of rice leaf disease using the CNN algorithm which aims to facilitate farmers in detecting rice plant diseases quickly and accurately through image analysis.

In research conducted by Fakhri Habib Hawari, Faslah Fadillah, Muhamad Rifqi Alviandi, and Toni Arifin in 2022 discusses the use of the CNN (Convolutional Neural Network) algorithm in classifying diseases in rice plants. The CNN algorithm has proven superior in various applications in the real world. The purpose of this research is to assist farmers in identifying and classifying diseases in rice plants, as well as reducing the risk of crop failure due to leaf diseases. This research covers disease types such as Brown Spot, Blight, Leaf Brown, and Healthy Leaf. The research process involves literature study, dataset collection, data preprocessing, and data processing. The data used for training, testing, and validation were obtained from various sources. The CNN algorithm was implemented using convolution layer (Conv2D), pooling layer (MaxPooling2D), flatten layer (flatten), and dense layer (Dense) for disease classification in rice leaves. The data training process is carried out as many as 10 epochs, and this process is stopped when the specified conditions are met. The results showed that the Deep Learning CNN method can be used to identify images of rice leaves affected by disease. The highest accuracy of the training data reached 85%, the testing data reached 86%, and the validation data reached 95%. Therefore, this method is considered good enough in identifying images of disease in rice plants [18].

The second research conducted by Amanda Caecilia Milano, Achmad Yasid, and Rima Tri Wahyuningrum in 2024 on the Classification of Rice Leaf Diseases Using the Efficientnet-B6 Deep Learning Model. The study explains the Convolutional Neural Network method using the EfficientNet-B6 architecture is a method that can be used for image classification or images which can help to find out the disease of rice plants and carry out further prevention and maintenance of rice plants. This study uses 3355 images of rice plant leaves which are divided into four classes, namely Healthy as many as 1488 images, LeafBlast as many as 779 images, Hispa as many as 565 images, and BrownSpot as many as 523 images. Classification is carried out using the Deep Learning method with EfficientNet-B6 architecture and the best performance results are obtained in the scenario with the input size value = 224 and the number of epochs = 50, the highest accuracy results are in the fifth fold with an accuracy value of 77.05% [19].

The third research conducted by Rahma Shinta, Jasril, Muhammad Irsyad, Febi Yanto, and Suwanto Sanjaya in 2023 on Image Classification of Rice Plant Leaf Diseases Using CNN with VGG-19 Architecture. This research applies the Convolutional Neural Network (CNN) method with VGG-19 architecture for the classification of leaf disease images of rice plants. The purpose of this research is to compare the test accuracy results of models that use augmentation and without data augmentation. The data in this study is divided into 4 classes, namely blast, brown spot, leaf smut, and healthy with the original data of 440 and augmented data of 1320 images. The test results show that the highest accuracy using data augmentation obtained is 94.31%, while the highest accuracy without data augmentation obtained is 93.18%. The results show that augmentation can improve accuracy results. The use of the Nadam optimizer produces a higher accuracy value than Adamax. Hyper parameters used also affect the test accuracy results.

Copyright © 2024 **Author**, Page 1294
This Journal is licensed under a Creative Commons Attribution 4.0 International License

**Building of Informatics, Technology and Science (BITS)**
Volume 6, No 3, Desember 2024 Page: 1293−1300
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v6i3.5883

## 2. RESEARCH METHODOLOGY

The methodology used in this research is the Convolution Neural Network (CNN) method to complete the classification of diseases in rice leaves using images of rice leaves, the following is a picture of the classification research stages.

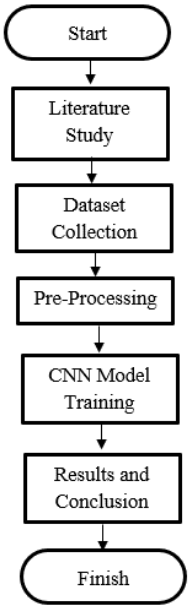


**Figure 1.** Classification method using CNN

The following is an explanation in figure 1. Classification method using CNN:

a. Literature Study
Literature study is to explore knowledge and references from various books and journals that have a relationship between the research that will be carried out. The purpose of the literature study is as an auxiliary material for conducting research by studying previous research sources [20]. Literature study is a reference taken from journals and websites (online research) related to deep learning, plant disease classification, and CNN.

b. Dataset Collection
Data collection uses a general dataset from Kaggle.com which is a website that provides data for machine learning projects. The dataset taken in the form of 3355 image files which are divided into 4 classes namely Healthy, hispa, brownspot, leafblast.

c. Pre-Processing
Data preprocessing is the preparation stage before the data is processed and used for classification [18]. Preprocessing process starts with crop image, image augmentation, and split dataset into three, namely training data, validation data, and testing data.

d. CNN Model Training
At this stage the author trains the model that has been made in Google Colab using the Convolutional Neural Network (CNN) algorithm for the classification of rice leaf diseases using convolution layers (Conv2D) [21], layer pooling (MaxPooling2D), layer flatten (Flatten), and layer dense (Dense) [22]. Then the model that has been made is trained using a website made by Google, namely teachablemachine. This website can train dataset models automatically, to create image models that will later be used to detect rice plant diseases in the form of websites [23].

e. Result and Conclusion
Results and conclusions are the last stage in this research. Where at this stage the results of the accuracy percentage of the CNN model that has been made will be obtained, and stored in the Hierarchical Data (HDF) format with the .h5 extension which will later be used to build a classification web.

## 3. RESULT AND DISCUSSION

This stage discusses a series of processes and analyzes the results of the classification of rice diseases through leaf images using the Convolutional Neural Network (CNN) algorithm. In this research, the author took several steps including:

Copyright © 2024 **Author**, Page 1295
This Journal is licensed under a Creative Commons Attribution 4.0 International License

**Building of Informatics, Technology and Science (BITS)**
Volume 6, No 3, Desember 2024 Page: 1293−1300
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v6i3.5883

### 3.1 Literature Study

Literature study is a reference taken from journals and websites (online research) related to deep learning, classification of rice plant diseases, CNN.

### 3.2 Dataset Collection and Sources

Data collection uses a common dataset derived from Kaggle.com with the URL: https://www.kaggle.com/code/ratul6/rice-leafs-detection/data. The dataset contains images of rice leaves that are used for the classification process. The dataset taken is an image with a total dataset of 3355 images of rice plant leaves which are divided into four classes, namely Healthy as many as 1488 images, LeafBlast as many as 779 images, Hispa as many as 565 images, and BrownSpot as many as 523 images.


**Figure 2.** Rice Plant Diseases

Figure 2., is a type of rice plant disease that is divided into 4 classes, namely Healthy or healthy types of rice plants with a total data of 1488 images, Leafblast or also called a disease that has yellow spots on the tip with a total data of 779 images, Brownspot or also called oval-shaped brown spots that are evenly distributed on the surface of the leaves with a gray or white center point with a total data of 523 images, and Hispa or also called a disease that has large white spots due to adult insect attacks that erode the surface of the leaves with a total data of 565 images.

### 3.3 Pre–Processing

Importing the library used first, here using the TensorFlow library is the most popular and widely used library for developing and implementing Machine Learning and other algorithms that have many mathematical operations to perform. Then the matplotlib library to plot graphs and display images in training and validation data, numpy is used to convert python lists to numpy arrays and perform the necessary matrix operations. and the os library is used to read files and directory structures.

```
[ ]  #import liobraries
     import tensorflow as tf
     from tensorflow.keras import models, layers
     import matplotlib.pyplot as plt
     import numpy as np
     import os
```

```
[ ]  IMAGE_SIZE = 256
     BATCH_SIZE = 32
     CHANNELS = 3
     EPOCHS = 300
```

**Figure 3.** Library

In Figure 3., the Import tensorflow code is used to import or activate the tensorflow library, Import matplotlib.pyplot is used to import or activate the matplotlib pyplot library, Import numpy is used to import or activate the numpy library, and Import os is used to import or activate the operating system's built-in function library and read our directory in python.

prepares the image data before it is fed into the deep learning model. The process of resizing and scaling ensures the images are of a consistent size and scale, while data augmentation helps to enrich the variety of data to make the model more robust and resilient to small changes in the input images.

```
[ ]  #resize and rescale
     resize_and_rescale = tf.keras.Sequential([
         layers.experimental.preprocessing.Resizing(IMAGE_SIZE,IMAGE_SIZE), #resize new inputs
         layers.experimental.preprocessing.Rescaling(1.0/255) #scale down RGB
     ])
```

```
[ ]  #data_augmentation
     data_augmentation = tf.keras.Sequential([
         layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
         layers.experimental.preprocessing.RandomRotation(0.2),
     ])
```

**Figure 4.** Pre-Processing Dataset

Copyright © 2024 **Author**, Page 1296
This Journal is licensed under a Creative Commons Attribution 4.0 International License

**Building of Informatics, Technology and Science (BITS)**
Volume 6, No 3, Desember 2024 Page: 1293−1300
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v6i3.5883

Figure 4., shows two sections of Python code that serve for image preprocessing in the machine learning model using TensorFlow and Keras. The first part is the resize and rescale process that aims to resize and normalize the image before it is inserted into the model. This process is performed using the tf.keras.Sequential object, which contains two layers of preprocessing. The first layer, Resizing, resizes the input images to the desired dimensions (IMAGE_SIZE x IMAGE_SIZE), ensuring that all processed images are of consistent size. The second layer, Rescaling, rescales the pixel values from 0-255 to 0-1 by dividing each pixel value by 255, which aims to speed up model training and ease convergence. While the second part is data augmentation, which is used to expand the variety of training data to improve the generalization ability of the model. In this data augmentation, two techniques are applied: first, RandomFlip which randomly flips the image both horizontally and vertically, and second, RandomRotation which randomly rotates the image with a rotation angle of up to 0.2 radians. This augmentation allows the model to learn more data variations without having to manually increase the number of datasets, thus making the model more robust in dealing with image variations in future test data.

### 3.4 CNN Model Training

Then create a Convolutional Neural Network (CNN) Algorithm model architecture for rice leaf disease classification using convolution layer (Conv2D), pooling layer (MaxPooling2D), flatten layer (Flatten), and dense layer (Dense). The activation function used is ReLu. The kernel/filter size used varies for the convolution layer which is 3x3, 2x2. While the pooling size used is 2x2. the number of filters/kernels used also varies, namely 32 filters and 64 filters [21]. Dataset scenario with 80% train data comparison, 10% validation data test 10% with Adam optimizer and 300 epoch value.

```
[ ]  input_shape = (BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE, CHANNELS)
     n_classes = 4

     model = models.Sequential([
         resize_and_rescale,
         data_augmentation,
         layers.Conv2D(32, (3,3), activation='relu', input_shape = input_shape), #google Conv2D for all arguement. 32=no. of layers
         layers.MaxPooling2D((2,2)),
         layers.Conv2D(64, kernel_size=(3,3), activation='relu'),
         layers.MaxPooling2D((2,2)),
         layers.Conv2D(64, kernel_size=(3,3), activation='relu'),
         layers.MaxPooling2D((2,2)),
         layers.Conv2D(64,(3,3), activation='relu'),
         layers.MaxPooling2D((2,2)),
         layers.Conv2D(64,(3,3), activation='relu'),
         layers.MaxPooling2D((2,2)),
         layers.Conv2D(64,(3,3), activation='relu'),
         layers.MaxPooling2D((2,2)),
         #flatten into an array of neurons
         layers.Flatten(),
         layers.Dense(64, activation='relu'), #dense layer of 64 neurons
         layers.Dense(n_classes, activation='softmax'), #softmax normalizes the prob of classes.
     ])

     model.build(input_shape=input_shape)
```

**Figure 5.** CNN Model

Figure 5., shows the Python code used to build the convolutional neural network (CNN) model using the TensorFlow framework and Keras. The model is designed for the task of classifying images into four classes. Initially, the code defines input_shape, which is the shape of the input data including batch size, image dimensions, and number of channels. The model is built using models.Sequential, which means each layer is added sequentially. The model starts with the first convolution layer which has 32 filters, a 3x3 kernel, and a ReLU activation function. The input shape is specified by input_shape, and this layer is followed by a pooling layer to reduce the dimensionality of the data. Next, there are multiple convolution layers that have 64 filters with the same kernel size, and each convolution layer is followed by a pooling layer to further reduce the dimensionality of the data. After a series of convolution and pooling layers, the data is flattened to one dimension using the Flatten() layer. This allows the data to be processed by the dense layer which has 64 neurons with ReLU activation. The last layer is a dense layer with the number of neurons corresponding to the number of target classes, using softmax activation to convert the output into probabilities for each class. Finally, the model is built by calling the model.build method which initializes the model based on a predefined form of input. The model is designed to receive images, process them through convolution and pooling layers, and finally classify them into one of the four classes.

This architecture is designed to maximize the model's ability to recognize and classify diseases in rice plant images with high accuracy through a combination of various image processing techniques and artificial neural networks.

Copyright © 2024 **Author**, Page 1297
This Journal is licensed under a Creative Commons Attribution 4.0 International License

**Building of Informatics, Technology and Science (BITS)**
Volume 6, No 3, Desember 2024 Page: 1293−1300
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v6i3.5883

### 3.5 Result And Conclusion

```
[ ] #good to record the history of every epochs in params
    history = model.fit(
        train_ds,
        epochs=EPOCHS,
        batch_size=BATCH_SIZE,
        verbose=1,
        validation_data=val_ds
    )
```

```
Epoch 1/300
84/84 [==============================] - 862s 152ms/step - loss: 1.3011 - accuracy: 0.4417 - val_loss: 1.2815 - val_accuracy: 0.4250
Epoch 2/300
84/84 [==============================] - 6s 71ms/step - loss: 1.2352 - accuracy: 0.4584 - val_loss: 1.1873 - val_accuracy: 0.4875
Epoch 3/300
84/84 [==============================] - 6s 71ms/step - loss: 1.1948 - accuracy: 0.4935 - val_loss: 1.2039 - val_accuracy: 0.4906
Epoch 4/300
84/84 [==============================] - 6s 71ms/step - loss: 1.1794 - accuracy: 0.5091 - val_loss: 1.2107 - val_accuracy: 0.4938
Epoch 5/300
84/84 [==============================] - 6s 71ms/step - loss: 1.1723 - accuracy: 0.5039 - val_loss: 1.2018 - val_accuracy: 0.4969
Epoch 6/300
84/84 [==============================] - 6s 71ms/step - loss: 1.1541 - accuracy: 0.5170 - val_loss: 1.1650 - val_accuracy: 0.5125
Epoch 7/300
84/84 [==============================] - 6s 71ms/step - loss: 1.1384 - accuracy: 0.5147 - val_loss: 1.1624 - val_accuracy: 0.5094
Epoch 8/300
84/84 [==============================] - 6s 70ms/step - loss: 1.1433 - accuracy: 0.5162 - val_loss: 1.1521 - val_accuracy: 0.5125
Epoch 9/300
84/84 [==============================] - 6s 70ms/step - loss: 1.1197 - accuracy: 0.5177 - val_loss: 1.1578 - val_accuracy: 0.5094
Epoch 10/300
84/84 [==============================] - 6s 71ms/step - loss: 1.1390 - accuracy: 0.5162 - val_loss: 1.2766 - val_accuracy: 0.4625
Epoch 11/300
84/84 [==============================] - 6s 71ms/step - loss: 1.1289 - accuracy: 0.5132 - val_loss: 1.1286 - val_accuracy: 0.5188
Epoch 12/300
84/84 [==============================] - 6s 71ms/step - loss: 1.1191 - accuracy: 0.5177 - val_loss: 1.1183 - val_accuracy: 0.5031
Epoch 13/300
84/84 [==============================] - 6s 70ms/step - loss: 1.1051 - accuracy: 0.5311 - val_loss: 1.0884 - val_accuracy: 0.5312
Epoch 14/300
84/84 [==============================] - 6s 71ms/step - loss: 1.0833 - accuracy: 0.5464 - val_loss: 1.1269 - val_accuracy: 0.4969
Epoch 286/300
84/84 [==============================] - 6s 72ms/step - loss: 0.3253 - accuracy: 0.8815 - val_loss: 0.4135 - val_accuracy: 0.8625
Epoch 287/300
84/84 [==============================] - 6s 72ms/step - loss: 0.3430 - accuracy: 0.8729 - val_loss: 0.4429 - val_accuracy: 0.8531
Epoch 288/300
84/84 [==============================] - 6s 73ms/step - loss: 0.3715 - accuracy: 0.8576 - val_loss: 0.4191 - val_accuracy: 0.8500
Epoch 289/300
84/84 [==============================] - 6s 72ms/step - loss: 0.3172 - accuracy: 0.8796 - val_loss: 0.4155 - val_accuracy: 0.8594
Epoch 290/300
84/84 [==============================] - 6s 73ms/step - loss: 0.3158 - accuracy: 0.8792 - val_loss: 0.4198 - val_accuracy: 0.8625
Epoch 291/300
84/84 [==============================] - 6s 72ms/step - loss: 0.3234 - accuracy: 0.8736 - val_loss: 0.5655 - val_accuracy: 0.8188
Epoch 292/300
84/84 [==============================] - 6s 72ms/step - loss: 0.3301 - accuracy: 0.8818 - val_loss: 0.4282 - val_accuracy: 0.8625
Epoch 293/300
84/84 [==============================] - 6s 73ms/step - loss: 0.3364 - accuracy: 0.8733 - val_loss: 0.3633 - val_accuracy: 0.8969
Epoch 294/300
84/84 [==============================] - 6s 73ms/step - loss: 0.3233 - accuracy: 0.8852 - val_loss: 0.4434 - val_accuracy: 0.8813
Epoch 295/300
84/84 [==============================] - 6s 72ms/step - loss: 0.3175 - accuracy: 0.8811 - val_loss: 0.4702 - val_accuracy: 0.8656
Epoch 296/300
84/84 [==============================] - 6s 73ms/step - loss: 0.3477 - accuracy: 0.8748 - val_loss: 0.4546 - val_accuracy: 0.8813
Epoch 297/300
84/84 [==============================] - 6s 73ms/step - loss: 0.3031 - accuracy: 0.8897 - val_loss: 0.4119 - val_accuracy: 0.8906
Epoch 298/300
84/84 [==============================] - 6s 73ms/step - loss: 0.3039 - accuracy: 0.8871 - val_loss: 0.5754 - val_accuracy: 0.8375
Epoch 299/300
84/84 [==============================] - 6s 73ms/step - loss: 0.3615 - accuracy: 0.8602 - val_loss: 0.4573 - val_accuracy: 0.8656
Epoch 300/300
84/84 [==============================] - 6s 73ms/step - loss: 0.2964 - accuracy: 0.8856 - val_loss: 0.4469 - val_accuracy: 0.8781
```

```
[ ] #test model with test_ds
    scores = model.evaluate(test_ds)
```

```
11/11 [==============================] - 6s 32ms/step - loss: 0.4166 - accuracy: 0.8608
```

**Figure 6.** Results from CNN Model

Figure 6., shows the results of training the neural network model for rice plant disease classification for 300 epochs. Each epoch records the loss and accuracy for training and validation data. At the end of training, the model achieved training accuracy of 88.56% and validation accuracy of 87.81%, with training loss of 0.2964 and validation loss of 0.4469. The model was then evaluated on the test dataset, resulting in a test accuracy of 86.08% and a test loss of 0.4166. Overall, the model showed good stability with no significant signs of overfitting, but there was a slight discrepancy between training and test accuracies, indicating potential for further improvement in terms of model generalization, such as by using regularization techniques or better hyperparameter tuning.

Copyright © 2024 **Author**, Page 1298
This Journal is licensed under a Creative Commons Attribution 4.0 International License

**Building of Informatics, Technology and Science (BITS)**
Volume 6, No 3, Desember 2024 Page: 1293−1300
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v6i3.5883

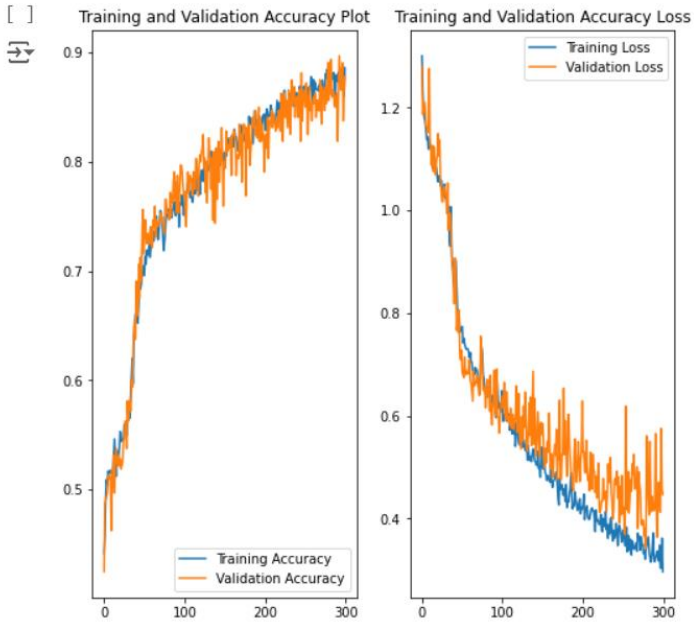


**Figure 7.** Plot Result

Figure 7., shows two plots that illustrate the performance of the model during the training process in terms of accuracy and loss. In the plot on the left, we see a graph of Training and Validation Accuracy over 300 epochs. The blue line represents training accuracy, while the orange line represents validation accuracy. Both lines show a consistent upward trend, with both training accuracy and validation accuracy increasing with each epoch. At the beginning of training, there are large fluctuations, but over time, the graph becomes more stable and reaches an accuracy close to 90% by the end of training. The lines that are close to each other indicate that the model is not significantly overfitting, as the validation accuracy still follows the trend of the training accuracy with a small difference. Meanwhile, in the plot on the right, the Training and Validation Loss graphs are shown. The blue line shows the training loss, and the orange line shows the validation loss. Both losses tend to decrease as the epochs increase, which is an indication that the model is learning well. However, the validation loss is slightly more volatile than the training loss, which could be a sign that the model is struggling a bit with generalization to data that was not seen during training. However, the overall decrease shows that the model is successfully reducing the prediction error, and the stabilization of the loss value around the 300th epoch indicates that the model has reached a fairly optimal performance. So overall, this graph shows that the model is experiencing a steady improvement in performance on both training and validation data, with increasing accuracy and decreasing loss, although there is a slight fluctuation in validation loss.

```
[ ] model_version = 1
    model.save(f"rice_models.h5/{model_version}")

[ ] model.save('keras_model.h5')

/usr/local/lib/python3.10/dist-packages/keras/src/en
  saving_api.save_model(
```

**Figure 8.** Save CNN Model

Figure 8., explains that the model can be saved in .h5 format. The .h5 format is the HDF5 (Hierarchical Data Format version 5) file format, which is used to store data in a structured form.

## 4. CONCLUSION

This research has successfully created a rice plant disease classification model using Convolutional Neural Network (CNN) with 4 types of diseases, namely brownspot, hispa, healthy, and leafblast. After being trained for 300 epochs, the model achieved training accuracy of 88.56% and validation accuracy of 87.81%, with training loss of 0.2964 and validation loss of 0.4469. Evaluation of the model on the test dataset showed a test accuracy of 86.08% and a test loss of 0.4166, which indicates that the model is able to identify rice leaf diseases quite well.

Copyright © 2024 **Author**, Page 1299
This Journal is licensed under a Creative Commons Attribution 4.0 International License

**Building of Informatics, Technology and Science (BITS)**
Volume 6, No 3, Desember 2024 Page: 1293−1300
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v6i3.5883

# REFERENCES

[1] R. Shinta, Jasril, M. Irsyad, F. Yanto, and S. Sanjaya, "Klasifikasi Citra Penyakit Daun Tanaman Padi Menggunakan CNN dengan Arsitektur VGG-19," *J. Sains dan Inform.*, vol. 9, no. 1, pp. 37–45, 2023, doi: 10.22216/jsi.v9i1.2175.

[2] H. P. Angjaya, K. Gunadi, and R. Adipranata, "Pengenalan Penyakit pada Tanaman Pokok di Indonesia dengan Metode Convolutional Neural Network," *J. Infra*, 2021.

[3] Herwina, Darmatasia, A. K. Ash Shiddiq, and T. D. Syahputra, "Deteksi Penyakit pada Tanaman Padi Menggunakan MobileNet Transfer Learning Berbasis Android," vol. 2, no. 2, 2022, doi: https://doi.org/10.24252/jagti.v2i2.41.

[4] S. Yuliany, Aradea, and Andi Nur Rachman, "Implementasi Deep Learning pada Sistem Klasifikasi Hama Tanaman Padi Menggunakan Metode Convolutional Neural Network (CNN)," *J. Buana Inform.*, vol. 13, no. 1, pp. 54–65, 2022, doi: 10.24002/jbi.v13i1.5022.

[5] M. G. Lanjewar and K. G. Panchbhai, "Convolutional neural network based tea leaf disease prediction system on smart phone using paas cloud," *Neural Comput. Appl.*, vol. 35, no. 3, pp. 2755–2771, 2023, doi: 10.1007/s00521-022-07743-y.

[6] W. B. Demilie, "Plant disease detection and classification techniques: a comparative study of the performances," *J. Big Data*, vol. 11, no. 1, 2024, doi: 10.1186/s40537-023-00863-9.

[7] M. Khoiruddin, A. Junaidi, and W. A. Saputra, "Klasifikasi Penyakit Daun Padi Menggunakan Convolutional Neural Network," *Data Inst. Teknol. Telkom Purwokerto*, vol. 2, no. 1, pp. 37–45, 2022, doi: https://doi.org/10.20895/dinda.v2i1.341.

[8] M. Nabila, R. Idmayanti, and I. Rahmayuni, "Deteksi Wajah Bermasker Menggunakan Webcam dan AWS EC2 Berbasis Raspberry Pi," *JITSI J. Ilm. Teknol. Sist. Inf.*, vol. 2, no. 4, pp. 124–133, 2021, doi: 10.30630/jitsi.2.4.54.

[9] K. Rozi, Muhsi, Anwari, and T. M. Badri, "Klasifikasi Penyakit Tanaman Padi Berbasis Citra Daun Menggunakan Convolutional Neural Network (CNN)," *JSTIE (Jurnal Sarj. Tek. Inform.*, vol. 12, no. 1, p. 18, 2024, doi: 10.12928/jstie.v12i1.27314.

[10] M. F. R. MAULA, "Automated Valuation Model Untuk Estimasi Nilai Pasar Rumah Berbasis Jaringan Saraf Tiruan Backpropagation," 2022, doi: https://doi.org/10.33005/sibc.v16i2.28.

[11] M. S. Mubarok and M. I. Herdiansyah, "Implementasi Cloud Computing Amazon Web Services (AWS) Pada Web Reservasi Kamar Hotel," *Kaji. Ilm. Inform. dan Komput.*, vol. 4, no. 2, pp. 698–708, 2023, doi: 10.30865/klik.v4i2.1212.

[12] R. H. I. Sari and B. T. Handoko, "Sejarah web service dan Implementasi pada perusahaan Amazon Implementasi Algoritma RSA dan kriptografi Quantum pada Sistem Login View project WEB SERVICE View project," no. March, pp. 1–5, 2020.

[13] F. Muhammad, R. Saedudin, and A. Almaarif, "Analisis Performansi Metrik CPU Dan Memory Pada Windows Azure Virtual Machine ( VM ) dan Amazon Web Service Elastic Compute Cloud ( EC2 )," *eProceedings*, vol. 7, no. 2, pp. 6975–6983, 2020.

[14] Amazon Web Services, "Amazon EC2," 2023. [Online]. Available: https://aws.amazon.com/id/ec2/getting-started/#:~:text=Amazon Elastic Compute Cloud (Amazon,dimulai dengan cepat dan mudah.

[15] Amazon Web Services, "Amazon S3," 2023. [Online]. Available: https://aws.amazon.com/id/s3/?nc=sn&loc=0.

[16] Amazon Web Services, "Mengelola Peran IAM," 2023. [Online]. Available: https://aws.amazon.com/id/iam/features/manage-roles/.

[17] E. Anggiratih, S. Siswanti, S. K. Octaviani, and A. Sari, "Klasifikasi Penyakit Tanaman Padi Menggunakan Model Deep Learning Efficientnet B3 dengan Transfer Learning," *J. Ilm. SINUS*, vol. 19, no. 1, p. 75, 2021, doi: 10.30646/sinus.v19i1.526.

[18] F. H. Hawari, F. Fadillah, M. R. Alviandi, and T. Arifin, "Klasifikasi Penyakit Tanaman Padi Menggunakan Algoritma Cnn (Convolutional Neural Network)," *J. Responsif Ris. Sains dan Inform.*, vol. 4, no. 2, pp. 184–189, 2022, doi: 10.51977/jti.v4i2.856.

[19] A. C. Milano, A. Yasid, and R. T. Wahyuningrum, "Klasifikasi Penyakit Daun Padi Menggunakan Model Deep Learning Efficientnet-B6," *J. Inform. dan Tek. Elektro Terap.*, vol. 12, no. 1, 2024, doi: 10.23960/jitet.v12i1.3855.

[20] A. Jinan and B. H. Hayadi, "Klasifikasi Penyakit Tanaman Padi Mengunakan Metode Convolutional Neural Network Melalui Citra Daun (Multilayer Perceptron)," *J. Comput. Eng. Sci.*, vol. 1, no. 2, pp. 37–44, 2022.

[21] G. Y. Christiawan, R. A. Putra, A. Sulaiman, E. Poerbaningtyas, and S. W. Putri Listio, "Penerapan Metode Convolutional Neural Network (CNN) Dalam Mengklasifikasikan Penyakit Daun Tanaman Padi," *J-Intech*, vol. 11, no. 2, pp. 294–306, 2023, doi: 10.32664/j-intech.v11i2.1006.

[22] A. Khan, U. Nawaz, A. Ulhaq, and R. W. Robinson, "Real-time plant health assessment via implementing cloud-based scalable transfer learning on AWS DeepLens," *PLoS One*, vol. 15, no. 12 December, pp. 1–23, 2020, doi: 10.1371/journal.pone.0243243.

[23] B. A. Putra, A. P. Kharisma, and F. Al Huda, "Penelitian Akurasi Diagnosa Penyakit Tanaman Padi menggunakan Kamera dengan Metode Klasifikasi Gambar pada Perangkat Bergerak Android,"*Inf. dan Ilmu Komput. e-ISSN*, vol. 6, no. 10, 2022.

Copyright © 2024 **Author**, Page 1300
This Journal is licensed under a Creative Commons Attribution 4.0 International License