The Classification of Tea Leaf Disease Using CNN Image Classifier

Eliana Aida Rosyidah(✉), Alfian Futuhul Hadi, and Yuliani Setia Dewi

Department of Mathematics, University of Jember, Jember, Indonesia
elianarosyidah@gmail.com, {afhadi,yulidewi.fmipa}@unej.ac.id

Abstract. Technological developments have encouraged advances in plant disease control. One plant disease that needs to be controlled quickly is tea leaf disease. Tea leaf diseases are numerous. In this study, we took 4 disease samples. Image recognition of tea leaf diseases with Convolutional Neural Networks (CNN) is necessary for classification and disease control. The disease recognition process goes through an image classification process with a classification algorithm that has been pre trained with pre-labeled image data. A good model is essential for tea leaf disease recognition. A good model can be measured by accuracy, precision, and recall in the testing process. There are several factors that effect of the model, including the dataset division ratio, the number of repetitions (epoch). In this study, a dataset of 400 data was used which was divided into 4 classes to be trained namely algal leaf, anthracnose, bird eye spot, and healthy. Based on the results of the testing process, the best accuracy of 100%, precision of 95% and recall of 24% by applying of augmentation, precision and recall in the pre-processing stage, using a dataset ratio of 70:30 and epoch 150 has an effect in increasing the accuracy value.

Keywords: technological developments · tea leaf diseases · classification · convolutional neural networks · epochs

1 Introduction

According to statistical data in 2019 it shows that Indonesia is a tea producer seventh largest in the world. The development of tea leaf production in Indonesia from 2018 to 2020 tends to decline. There are problems that are often faced by farmers as including declining exports and imports, reduction in farmland and plant diseases. These tea leaf diseases can cause the poor growth of tea plant, resulting in a decrease in the yield and quality of tea leaves [1]. There are many tea leaf diseases as algal leaf, anthracnose, and bird eye spot.

Tea plant diseases are usually caused by fungi, bacteria, alga, and viruses. Also, there are other diseases which are caused by adverse environmental condition [2]. There are many characteristics of tea leaf disease. It takes a way to deal with diseases in tea leaves. There are disease recognition, classification, prevention and control.

I. H. Agustin (Ed.): ICONNSMAL 2022, AISR 177, pp. 89–112, 2023.
https://doi.org/10.2991/978-94-6463-174-6_10

Algal tea leaf disease is the disease affects the mature leaves that appear as circular green-gray areas that eventually form red rust as the alga produces a profusion of rust-colored microscopic spores [3]. Anthracnose is a disease caused by *Colletotrium theae sinensis*. The disease causes a decrease in the green color of the tea leaves. Bird eye spot caused by a fungal pathogen, *Cercospora theae*. Bird Eye Spot Disease affects mature tea leaves affecting photosynthesis and food supply.

Various methods are used to diagnose tea leaf disease. One of them with microscopic identification. However, the method takes a long time and can be subjective. So, need a faster way to deal with the problem. Accurate recognition and control of tea leaf diseases can maintain product quality and quantity.

The development of computing technology has been rapidly expanding in the field of agriculture. Image processing and machine learning can solve a variety of problems in agriculture. But, in recent years, deep learning is widely used for various image processing. Deep learning surpasses previous sophisticated machine learning. Deep learning is mainly used for image classification, object detection, natural language processing and automatically integrate feature extraction process into the training sector. Deep learning has succeeded in collection of large data sets and can be used to diagnose plant diseases quickly and accurately [4].

We propose a methodology consist of three main parts: data acquisition, pre-processing and classification. The method used is convolution neural network (CNN) by processing images for tea leaf disease classification. Based on the above description, this study aims to improve the accuracy in tea leaf disease classification by using different epoch values of 50, 100, and 150. The method we use is darknet-19 with ADAM hyperparameters with a learning rate of 0.1.

2 Method

The design of our proposed method is data sources, pre-processing and classification.

2.1 Dataset

Part of tea leaf disease images used in this paper is taken in Kaggle Tea Disease Leaf Dataset (PLD)|Kaggle. We selected several samples of tea leaf diseases. The data selected were infected images and healthy leaf images for comparison. The selected images take into consideration the various visual effects.

In this study we used 4 types is algal leaf, anthracnose, bird eye spot, and healthy. The images of various classes of tea leaf as follows (Fig. 1). There are 280 images in training dataset of 70 images in each class and 120 images in testing dataset of 30 images in each class. Each class has an equal number of training and testing dataset image. We used images augmentation to generate new images. Augmentation helps to build similar new images by rotating, flipping, cropping and resizing the existing images [5]. Image augmentation is used to prevent bias in the training process.

2.2 Training and Testing Data

After carrying out data augmentation, then the data is divided by the ratio of 70:30. Training data selected 70% randomly from total data and 30% testing from all data. The data used are 280 training data and 120 testing data. After going through training and testing data, 4 types of data will be generated namely FN, TN, FP and TP. This calculation uses confusion matrix to produce accuracy, recall and precision values of the model.

2.3 Artificial Neural Networks

Artificial intelligence (AI) is software capable of sophisticated, intelligent, computations similar to those that the human brain routinely performs [6]. Artificial intelligence simulates the workings of the human brain and deduces a set of rules, and models the workings of the brain called an ANN. In general, ANN are formed from millions of basic structures neutrons that are interconnected and integrated with each other so that can carry out activities regularly and continuously according to so that it can carry out activities regularly and continuously according to as needed. The formation of a neural network is created by an input layer, hidden layer and also output layer [7]. Can be seen in Fig. 1 is an illustration of a neural network design that has three layers.

If $W = (w_{ij})$ is a weight, namely the relationship between one neuron and another neuron. Network input to target unit Y_j (with no unit bias j) is a simple dot product of vectors $x = (x_1, x_2, \ldots, x_n)$ where n is the number of inputs dan w_j where j is the number of columns in the weight matrix. Thus, $\widehat{Y} = x \cdot w_j = \sum_{i=1}^{n} x_i w_{ij}$. Figure 2 illustrates model of artificial neural network. Bias (β) can be included in vector x by adding the component $x_0 = 1$, so the vector x becomes, $x = (1, x_1, x_2, \ldots, x_n)$.

Bias is considered like any other weight, i.e., $w_{0j} = \beta_j$. Network input to the unit Y_j is given as Eq. (1) below.

$$\widehat{Y} = \sum_{i=0}^{n} x_i w_{ij} = w_{0j} + \sum_{i=0}^{n} x_i w_{ij} = \beta_j + \sum_{i=0}^{n} x_i w_{ij} \tag{1}$$

There are several types of activation functions:

1. Linear activation function $\widehat{Y} = f(x) = ax + b$, Ketika $a = 1, b = 0$, itu adalah identitas

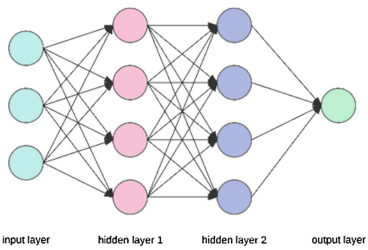


Fig. 1. Types of Multi Layers in Artificial Neural Networks.

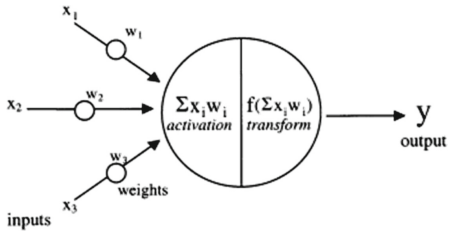


Fig. 2. Model of an Artificial Neural Networks.

2. Binary activation function with threshold (θ)

$$\widehat{Y} = f(x) = \begin{cases} 1 \ \textit{if} \ x \geq \theta \\ 0 \ \textit{if} \ x < \theta \end{cases}$$

3. Binary sigmoid activation function

$\widehat{Y} = f(x) = \frac{1}{1+e^{-\sigma x}}$, dan $f' = \sigma f(x)\big[1 - f(x)\big]$.

4. Bipolar sigmoid activation function

$\widehat{Y} = g(x) = 2f(x) - 1 = \frac{1-e^{-\sigma x}}{1+e^{-\sigma x}}$, dan $g'(x) = \frac{\sigma}{2}\big[1 + g(x)\big]\big[1 - g(x)\big]$.

5. Tangent hyperbolic activation function

$\widehat{Y} = h(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$, dan $h'(x) = [1 + h(x)][1 - h(x)]$.

- Backpropagation Algorithm

Backpropagation is a simple interconnection where if the output has a wrong result, the weights will be corrected so that the error will be minimized and close to the correct value.

Step 0. Initialize parameters
Weights (w), bias(β), learning rate (α), and.
error goal.
Step 1. Feedforward stage
$h_i = \beta + \sum_{i=0}^{n} x_i \cdot w_{i,j}$ where n is numbers of input.
Step 2. Calculate the output

$$y_{in_i} = \beta + \sum_{i=0}^{n} w_{i,j} \cdot h_i$$

Step 3. Output activation using log sigmoid

$$\hat{y}_m = \frac{1}{1 + e^{-y_{in_m}}}$$

Step 4. Calculate the output error

$$\delta k_i = \left(t - \hat{y}_i\right)^2$$

$$MSE = \frac{\sum_{i=0}^{n} \left(t - \hat{y}_i\right)^2}{n}$$

Step 5.Calculate the backpropagation error between output layer and hidden layer

$$\delta j_i^j = \beta \sum_{i=0}^{n} w_{i,j} * \delta k_i$$

Step 6.Calculate the backpropagation error between hidden layer and input layer

$$\delta j_i^j = \sum_{i=0}^{n} w_{i,j} * \delta k_i$$

Step 7. Update weight and bias

$$W_{new} = W_{old} + \alpha * \delta k_i * h_i$$

$$\beta_{new} = \beta_{old} + \alpha * \delta j_i + x_i$$

Step 8.Checking the termination criteria (epoch reaches the maximum number or $MSE \leq$ errorgoal).

Step 9. If not, go back to step 1

- Observation

Given a neural network with one hidden layer and two neurons. Figure 3 illustrates this artificial neural network. Suppose $x = \{x_i | i = 1, 2, \ldots, n\}$, where x is input data and n is numbers of input data. Let h dan β respectively be the number of hidden layers and bias. Output $\left(\hat{y}\right)$ written as $\hat{y} = \frac{1}{1+e^{-\left(\beta_1 + W^1\left(\beta_0 + W^0 x\right)\right)}}$, where $[h_1; h_2] = \beta^0 + W^0 x$.

Proof.

Input data $x = \{x_1, x_2, \ldots, x_n\}$, since we have one layer and two neurons, we define weights $W^0 = \begin{bmatrix} w_{1,1}^0 & w_{1,2}^0 & \cdots & w_{1,n}^0 \\ w_{2,1}^0 & w_{2,2}^0 & \cdots & w_{2,n}^0 \end{bmatrix}$, $W^1 = \begin{bmatrix} w_1^1 & w_2^1 \end{bmatrix}$ dan bias

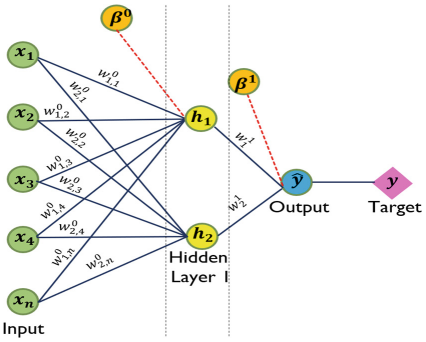


Fig. 3. Illustration of an ANN with One Hidden Layer.

β^0, β^1.

$$\begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} \beta_1^0 \\ \beta_2^0 \end{bmatrix} + \begin{bmatrix} w_{1,1}^0 & w_{1,2}^0 & \cdots & w_{1,n}^0 \\ w_{2,1}^0 & w_{2,1}^0 & \cdots & w_{2,n}^0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \beta^0 + W^0 x$$

Next, we get y_{in} as follow

$$y_{in} = \beta^1 + \begin{bmatrix} w_1^1 \\ w_2^1 \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = \beta^1 + \begin{bmatrix} w_1^1 \\ w_2^1 \end{bmatrix} \cdot \left(\beta^0 + W^0 x\right) = \beta^1 + W^1\left(\beta^0 + W^0 x\right)$$

So, output $\left(\hat{y}\right)$ written as follow

$$\hat{y} = \frac{1}{1 + e^{-y_{in}}} = \frac{1}{1 + e^{-(\beta^1 + W^1(\beta^0 + W^0 x))}}$$

It concludes the proof.

2.4 Convolutional Neural Networks

Convolutional Neural Networks have activation, bias and weight functions. Convolution Neural Network (CNN) is a deep learning algorithm that focuses on images and text. The input in CNN is a digital image, then processed based on the filter of each layer and produces a pattern from the image so that the classification process is easier. Below is the Convolutional Neural Network (CNN) architecture for tea leaf disease detection. This process consists of pre-processing stage, image extraction then classification. We created the CNN architecture as shown in Fig. 4.

Below is an illustration of the tea leaf disease classification process using Convolutional Neural Network. First, the image will go a pre-processing. Next, the input image will be extracted according to the illustration in the Fig. 5. Then the image size will be changed using zero padding, calculated convolution, ReLU, max pooling, smoothing, normalization, fully connected neural network and finally softmax.

- Splitting Image

All images need standardized. Images with large sizes will increases time complexity and produce large images so that resizing images is required. At this step, the size and

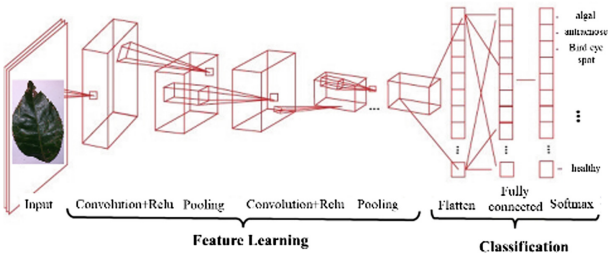


Fig. 4. The Architecture of Convolutional Neural Network.

scale of the image is changed, the dimensions are reduced and the image is cropped. Shown in Fig. 5.

- Feature Extraction

In order to extract deep features from pre-trained CNN, it is important to select a layer for each model. CNN models employ 2D convolution filters to analyze grayscale imagery and red, green, blue (RGB) and it can be seen Fig. 6.

- Zero Padding

Benefit of zero padding is to manage the output size. Illustration of adding zero padding can be seen in Fig. 7. By adding one zero padding, the output is exactly the same as the original input with N = 7, F = 3 and stride 1, the output will be 5x5.

- Convolution

Next is the convolution operation. Convolution layer is a layer that performs convolution. Convolution uses a certain size and performs matrix multiplication. The convolution operation aims to extract the features present in the input data. The filter shifts from the top left to the bottom right. This shift is called stride. Convolutional layer obtaining certain features. Convolution kernel parameters consist of kernel size, number, stride, padding, etc. These parameters should match the size of the original image. Convolution consists of red, green, and blue layers. The output of the convolution operation is called the feature map.

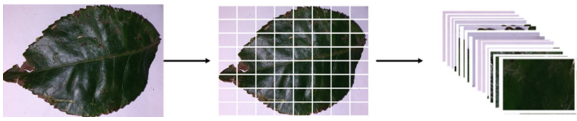


Fig. 5. Illustration of Splitting Process of Input Image.

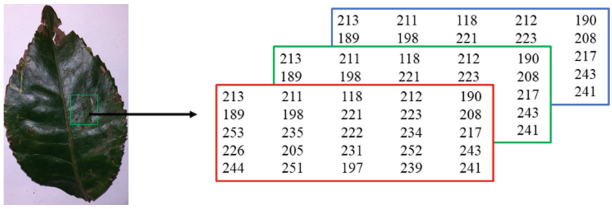


Fig. 6. Illustration of Input Image.

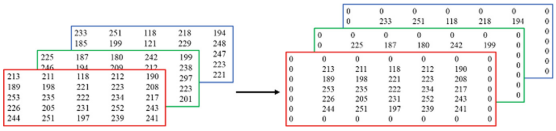


Fig. 7. Illustration of Added Zero Padding to the Color Channel Matrix.

Convolution operation in the red channel:

$$I_{red} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 213 & 211 \\ 0 & 189 & 198 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ -1 & 0 & -1 \end{bmatrix}$$

$$= (0 * 1) + (0 * 0) + \cdots + (198 * -1)$$

$$= 13$$

Convolution operation in the green channel:

$$I_{green} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 225 & 187 \\ 0 & 246 & 194 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & -1 \\ 0 & 0 & -1 \end{bmatrix}$$

$$= (0 * 1) + (0 * 0) + \cdots + (194 * -1)$$

$$= -156$$

Convolution operation in the blue channel:

$$I_{blue} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 233 & 251 \\ 0 & 185 & 199 \end{bmatrix} * \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

$$= (0 * -1) + (0 * 0) + \cdots + (199 * -1)$$

$$= -450$$

Suppose bias (β) = 1, The resulting feature map values are as follows:

$$c_1 = I_{red} + I_{green} + I_{blue} + \beta$$

$$= 13 - 156 - 450 + 1$$

$$= -592$$

The filter is then shifted one pixel to the right, so that the convolution operation is illustrated as follows:

Convolution operation in the red channel:

$$I_{red} = \begin{bmatrix} 0 & 0 & 0 \\ 213 & 211 & 118 \\ 189 & 198 & 221 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 1 \\ 1 & -1 & 1 \\ -1 & 0 & -1 \end{bmatrix}$$

$$= (0 * 1) + (0 * 0) + \cdots + (221 * -1)$$

$$= -79$$

Convolution operation in the green channel:

$$I_{green} = \begin{bmatrix} 0 & 0 & 0 \\ 225 & 187 & 180 \\ 246 & 194 & 209 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -1 \\ -1 & 0 & -1 \end{bmatrix}$$

$$= (0 * 1) + (0 * 0) + \cdots + (209 * -1)$$

$$= -202$$

Convolution operation in the blue channel:

$$I_{blue} = \begin{bmatrix} 0 & 0 & 0 \\ 233 & 251 & 118 \\ 185 & 199 & 121 \end{bmatrix} * \begin{bmatrix} -1 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & -1 \end{bmatrix}$$

$$= (0 * -1) + (0 * 0) + \cdots + (121 * -1)$$

$$= -287$$

Then, the resulting feature map values are as follows:

$$c_2 = I_{red} + I_{green} + I_{blue} + \beta$$

$$= -79 - 202 - 287 + 1$$

$$= -567$$

After going through a series of convolution processes, the following feature map is obtained:

$$C = \begin{bmatrix} c_1 & c_2 & \cdots & c_5 \\ c_6 & c_7 & \cdots & c_{10} \\ \vdots & \vdots & \ddots & \vdots \\ c_{21} & c_{22} & \cdots & c_{25} \end{bmatrix}$$

$$= \begin{bmatrix} -592 & -567 & -770 & -754 & 200 \\ -1 & 95 & 75 & -63 & 466 \\ -118 & 483 & 162 & 703 & 509 \\ -52 & 447 & 158 & 533 & 473 \\ 735 & 781 & 903 & 974 & 417 \end{bmatrix}$$

- Rectified Linear Unit (ReLU)

ReLU (Rectified Linear Unit) is different from other activation functions, all neurons are not activated at the same time. If the feature map is negative then ReLU will change it to zero and positive for neurons that are not activated or zero. The following is a feature map that has been activated with Rectified Linear Unit (ReLU).

$$C(\sigma) = \begin{cases} \sigma, & \text{if } \sigma \geq 0 \\ 0, & \text{if } \sigma < 0 \end{cases}$$

$$C = \begin{bmatrix} 0 & 0 & 0 & 0 & 200 \\ 0 & 95 & 75 & 0 & 466 \\ 0 & 483 & 162 & 703 & 509 \\ 0 & 447 & 158 & 533 & 473 \\ 735 & 781 & 903 & 974 & 417 \end{bmatrix}$$

- Polling Layer

Pooling is a combination that helps reduce data [9]. Max pooling is the maximum valued sub matrix. Before becoming a fully connected neural network (FCNN), the matrix dimension needs to be reduced using maxpooling. Larger pooling areas are too risky during training and smaller areas lead to over-fitting [10]. Max pooling preserves the maximum value by dividing the input. Where the filter will move and count from the top left corner to the bottom right with a shift of one pixel. And average pooling is the average value matrix. With the max pooling layer, the process will be accelerated because the data will be reduced to 75%. The following illustrates the MaxPooling process with 3x3 matrix.

$$m_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 95 & 75 \\ 0 & 483 & 162 \end{bmatrix}$$

$$\max(m_1) = 483$$

The filter shifting the right by one pixel, the second results of the MaxPooling on the filter are illustrated below.

$$m_2 = \begin{bmatrix} 0 & 0 & 0 \\ 95 & 75 & 0 \\ 483 & 162 & 703 \end{bmatrix}$$

$$\max(m_2) = 703$$

The filter continues to move to the bottom right. After the process is complete, the feature map is obtained as follows:

$$C = \begin{bmatrix} 483 & 703 & 703 \\ 483 & 533 & 533 \\ 903 & 974 & 974 \end{bmatrix}$$

- Flattening

The Flatten layer aims to convert the data into one dimension. Output the flatten is a single long feature vector as follows:

$$C = [483; 703; 703; 483; 533; 533; 903; 974; 974]$$

- Normalization

Normalization is used to homogenize the pixel value of the image later. This technique uses the calculation of standard deviation. Then, perform mini batch averaging. Next, perform mini batch variation and finally perform normalization. There are several steps that must be taken when performing data normalization. The first step is that the dataset must be sorted (training data, validation data, and test data) first before normalization. Then we normalize the vector C with Eq. 2.

$$x_i = \frac{a + (C_i - \max(C)) \times (b - a)}{\max(C) - \min(C)} \tag{2}$$

where: $a = 0.1; b = 0.9$

- Fully Connected Neural Network

The fully connected layer is a similar to the way that neurons are arranged in a traditional neural network [11]. This layer aims to transform the dimensions of the data which can later be classified linearly. So, we obtain the input vector for FCNN as follows.

$$x = [0.1; 0.45; 0.45; 0.1; 0.18; 0.18; 0.78; 0.9; 0.9]$$

These vectors then become input to Fully-Connected Neural Networks (FCNN). Some of the parameters we used in FCNN are weights $\left(W^0 \text{and} W^1\right)$, bias $\left(\beta^0 \text{and} \beta^1\right)$, learning rate $\alpha = 0.1$, and target (t).

$$W^0 = \begin{bmatrix} 0.30.40.10.10.30.40.30.20.1 \\ 0.10.30.20.40.40.20.20.30.1 \\ 0.20.20.20.30.10.40.30.10.2 \\ 0.40.10.20.20.10.40.10.10.2 \end{bmatrix}$$

$$W^1 = [0.3; 0.1; 0.2; 0.4]$$

$$\beta^0 = [0.1; 0.1; 0.1; 0; 1]$$

$$\beta^1 = [0.1; 0.1; 0.1; 0.1]$$

$$t = \begin{bmatrix} 1 & 1 & 11 \end{bmatrix}$$

The first step in FCNN is feedforward neuron one as follows.

$$h^1_{1,1} = \beta^0_1 + x_1 * w^0_{1,1} = 0.1 + 0.1 * 0.3 = 0.1300$$

$$h_{1,2}^1 = \beta_1^0 + x_2 * w_{1,2}^0 = 0.1 + 0.45 * 0.4 = 0.2800$$

$$\vdots$$

$$h_{1,9}^1 = \beta_1^0 + x_9 * w_{1,9}^0 = 0.1 + 0.9 * 0.1 = 0.1900$$

$$h_1^1 = \begin{bmatrix} 0.13 & 0.28 & \cdots & 0.19 \end{bmatrix}$$

Next is feedforward neuron two as follow

$$h_{2,1}^1 = \beta_2^0 + x_1 * w_{2,1}^0 = 0.1 + 0.1 * 0.1 = 0.1100$$

$$h_{2,2}^1 = \beta_2^0 + x_2 * w_{2,2}^0 = 0.1 + 0.45 * 0.3 = 0.2350$$

$$\vdots$$

$$h_{2,9}^1 = \beta_2^0 + x_9 * w_{2,9}^0 = 0.1 + 0.9 * 0.1 = 0.1900$$

$$h_2^1 = \begin{bmatrix} 0.11 & 0.235 & \cdots & 0.19 \end{bmatrix}$$

The last feedforward is in neuron three, the calculate as follow

$$h_{3,1}^1 = \beta_3^0 + x_1 * w_{3,1}^0 = 0.1 + 0.1 * 0.2 = 0.1200$$

$$h_{3,2}^1 = \beta_3^0 + x_2 * w_{3,2}^0 = 0.1 + 0.45 * 0.2 = 0.1900$$

$$\vdots$$

$$h_{3,9}^1 = \beta_3^0 + x_9 * w_{3,9}^0 = 0.1 + 0.9 * 0.2 = 0.1180$$

$$h_3^1 = \begin{bmatrix} 0.12 & 0.19 & \cdots & 0.118 \end{bmatrix}$$

The last feedforward is in neuron four, the calculate as follow

$$h_{4,1}^1 = \beta_4^0 + x_1 * w_{4,1}^0 = 0.1 + 0.1 * 0.4 = 0.1400$$

$$h_{4,2}^1 = \beta_4^0 + x_2 * w_{4,2}^0 = 0.1 + 0.45 * 0.1 = 0.1450$$

$$\vdots$$

$$h_{4,9}^{1} = \beta_4^0 + x_9 * w_{4,9}^0 = 0.1 + 0.9 * 0.2 = 0.1180$$

$$h_4^1 = \left[0.14\ 0.145\ \cdots 0.118 \right]$$

Next is to calculate the output (y)

$$yin_1 = \beta_1^1 + w_1^1 * h_{1,1}^1 + w_1^1 * h_{1,2}^1 + \cdots + w_1^1 * h_{1,9}^1$$

$$= 0.1 + 0.3 * 0.1300 + 0.3 * 0.2800 + \cdots + 0.3 * 0.1900$$

$$= 0.6385$$

$$yin_2 = \beta_2^1 + w_2^1 * h_{2,1}^1 + w_2^1 * h_{2,2}^1 + \cdots + w_2^1 * h_{2,9}^1$$

$$= 0.1 + 0.1 * 0.1100 + 0.1 * 0.2350 + \cdots + 0.1 * 0.1900$$

$$= 0.279$$

$$yin_3 = \beta_3^1 + w_3^1 * h_{3,1}^1 + w_3^1 * h_{3,2}^1 + \cdots + w_3^1 * h_{3,9}^1$$

$$= 0.1 + 0.2 * 0.1200 + 0.2 * 0.1900 + \cdots + 0.2 * 0.1180$$

$$= 0.3440$$

$$yin_4 = \beta_4^1 + w_4^1 * h_{4,1}^1 + w_4^1 * h_{4,2}^1 + \cdots + w_4^1 * h_{4,9}^1$$

$$= 0.1 + 0.4 * 0.1400 + 0.4 * 0.1450 + \cdots + 0.4 * 0.1180$$

$$= 0.5740$$

Then activate each neuron output. The activation function used is the sigmoid log.

$$y_m = \frac{1}{1 + e^{-yin_m}}$$

$$y_1 = 0.6544$$

$$y_2 = 0.5695$$

$$y_3 = 0.5852$$

$$y_4 = 0.6397$$

After obtaining the next output, we can calculate the output error

$$\delta k_1 = (t_1 - y_1)^2$$

$$= (1 - 0.6544)^2$$

$$= 0.1194$$

$$\delta k_2 = (t_2 - y_2)^2$$

$$= (1 - 0.5695)^2$$

$$= 0.1853$$

$$\delta k_3 = (t_3 - y_3)^2$$

$$= (1 - 0.5852)^2$$

$$= 0.1721$$

$$\delta k_4 = (t_4 - y_4)^2$$

$$= (1 - 0.6397)^2$$

$$= 0.1298$$

$$MSE = \frac{((t_1 - y_1) + (t_2 - y_2) + (t_3 - y_3) + (t_4 - y_4))^2}{4} = 0.6016$$

Then calculate the backpropagation error between output layer and hidden layer:

$$\delta j_1^0 = \beta_1^0 + w_1^1 * \Delta k_1$$

$$= 0.1 + 0.3 * 0.1194$$

$$= 0.1358$$

$$\delta j_2^0 = \beta_2^0 + w_2^1 * \Delta k_2$$

$$= 0.1 + 0.1 * 0.1853$$

$$= 0.1185$$

$$\delta j_3^0 = \beta_3^0 + w_3^1 * \Delta k_3$$

$$= 0.1 + 0.2 * 0.1721$$

$$= 0.1344$$

$$\delta j_4^0 = \beta_4^0 + w_4^1 * \Delta k_4$$

$$= 0.1 + 0.4 * 0.1298$$

$$= 0.1519$$

Then calculate the backpropagation error between hidden layer and input layer:

$$\delta j_1^1 = w_{1,1}^0 * \delta k_1 + w_{1,2}^0 * \delta k_1 + w_{1,3}^0 * \delta k_1 + \cdots + w_{1,9}^0 * \delta k_1$$

$$= 0.2627$$

$$\delta j_2^1 = w_{2,1}^0 * \delta k_2 + w_{2,2}^0 * \delta k_2 + w_{2,3}^0 * \delta k_2 + \cdots + w_{2,9}^0 * \delta k_2$$

$$= 0.4077$$

$$\delta j_3^1 = w_{3,1}^0 * \delta k_3 + w_{3,2}^0 * \delta k_3 + w_{3,3}^0 * \delta k_3 + \cdots + w_{3,9}^0 * \delta k_3$$

$$= 0.3442$$

$$\delta j_4^1 = w_{4,1}^0 * \delta k_4 + w_{4,2}^0 * \delta k_4 + w_{4,3}^0 * \delta k_4 + \cdots + w_{4,9}^0 * \delta k_4$$

$$= 0.2337$$

The final step is to update the weights and biases as follows:

- The weights between input layer and hidden layer

$$w_{new}{}_{1,1}^0 = w_{1,1}^0 + \alpha * \delta k_1 * h_1^1$$

$$= \left[0.3018\ 0.4038\ \cdots 0.1026 \right]$$

$$w_{new}{}^{0}_{2,1} = w^{0}_{2,1} + \alpha * \delta k_2 * h^{1}_{2}$$

$$= \left[0.1013\ 0.3028\ \cdots 0.1023 \right]$$

$$w_{new}{}^{0}_{3,1} = w^{0}_{3,1} + \alpha * \delta k_3 * h^{1}_{3}$$

$$= \left[0.2016\ 0.2016\ \cdots 0.1016 \right]$$

$$w_{new}{}^{0}_{4,1} = w^{0}_{4,1} + \alpha * \delta k_4 * h^{1}_{4}$$

$$= \left[0.4021\ 0.1022\ \cdots 0.1018 \right]$$

$$W_{new}{}^{0} = \begin{bmatrix} w_{new}{}^{0}_{1,1} \\ w_{new}{}^{0}_{2,1} \\ w_{new}{}^{0}_{3,1} \\ w_{new}{}^{0}_{4,1} \end{bmatrix}$$

$$W_{new}{}^{0} = \begin{bmatrix} 0.30180.4038 \ldots .0.1026 \\ 0.10130.3028 \ldots 0.1023 \\ 0.20160.2016 \ldots 0.1016 \\ 0.40210.1022 \ldots 0.1018 \end{bmatrix}$$

- The weights between hidden layer and output layer

$$w_{new}{}^{1}_{1} = w^{1}_{1} + \alpha * \delta k_1$$

$$= 0.3280$$

$$w_{new}{}^{1}_{2} = w^{1}_{2} + \alpha * \delta k_2$$

$$= 0.4446$$

$$w_{new}{}^{1}_{3} = w^{1}_{3} + \alpha * \delta k_3$$

$$= 0.1364$$

$$w_{new}{}^{1}_{4} = w^{1}_{4} + \alpha * \delta k_4$$

$$= 0.1249$$

$$W_{new}{}^{1} = \begin{bmatrix} w_{new}{}_{1}^{1} \\ w_{new}{}_{2}^{1} \\ w_{new}{}_{3}^{1} \\ w_{new}{}_{4}^{1} \end{bmatrix} = \begin{bmatrix} 0.3280 \\ 0.4446 \\ 0.1364 \\ 0.1249 \end{bmatrix}$$

- Bias between input layer and hidden layer

$$\beta_{new}{}_{1}^{0} = \beta_{1}^{0} + \alpha \cdot \delta \mathrm{j}_1 = 0.1136$$

$$\beta_{new}{}_{2}^{0} = \beta_{1}^{0} + \alpha \cdot \delta \mathrm{j}_2 = 0.1119$$

$$\beta_{new}{}_{3}^{0} = \beta_{1}^{0} + \alpha \cdot \delta \mathrm{j}_3 = 0.1134$$

$$\beta_{new}{}_{4}^{0} = \beta_{1}^{0} + \alpha \cdot \delta \mathrm{j}_4 = 0.1152$$

$$\beta_{new}{}^{0} = \left[\beta_{new}{}_{1}^{0}; \beta_{new}{}_{2}^{0}; \beta_{new}{}_{3}^{0}; \beta_{new}{}_{4}^{0}\right]$$

$$= [0.1136; 0.1119; 0.1134; 0.1152]$$

- Bias between hidden layer and output layer

$$\beta_{new}{}_{1}^{1} = \beta_{1}^{1} + \alpha \cdot \delta \mathrm{j}_1 = 0.1136$$

- Softmax Layer

The last fully connected layer is followed by a softmax which produces a probability distribution over the n class labels [10]. The softmax layer computes probability values due to its mathematical definition. This means that the sum f the output values is equal to 1. The values denote the degree of confidence on the result [8]. An illustration of the softmax process with output that matches the target shown in Fig. 8. While An illustration of the softmax process with output that matches the target shown in Fig. 9. The output value (logits) is processed using the softmax function. Based on the probability value generated, the system determines the third class as output.

2.5 Confusion Matrix (Metric Performance)

Confusion matrix shows the correct or incorrect prediction results in classification. The confusion matrix contains the actual classification results and the predicted classification system. System performance is usually evaluated using the data in the matrix. The table is presented in Table 1. Here is the performance of Confusion Matrix.

1) *True Positive (TP):* Is positive data that detected correctly.
2) *True Negative (TN):* Is the number of negative data detected correctly.
3) *False Positive (FP):* Is negative data but detected as positive data.

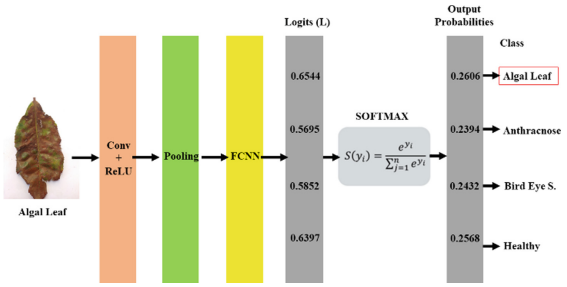


Fig. 8. Illustration Results from Softmax with Output that Matches Target.

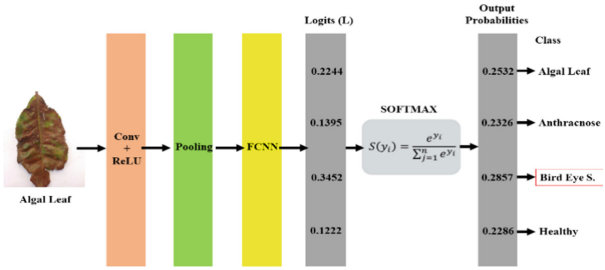


Fig. 9. Illustration Results from Softmax with Output that does not Matches Target.

4) *False Negative (FN):* Is the opposite of true positive, so that positive data is detected as negative data.

Accuracy, precision and recall values are obtained from TP, FP, FN, and TN. The accuracy compares the correct value with the overall value. Accuracy is calculated using Eq. (3).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% \tag{3}$$

Precision is the accuracy of the information with the answer given. Precision can be calculated using Eq. (4).

$$Precision = \frac{TP}{TP + FP} \times 100\% \tag{4}$$

Table 1. Confusion Matrix

	True	***False***
True	TP	FP
False	FN	TN

Table 2. Table Divided of Dataset

Class	**Data Set**	
	Training	***Testing***
Algal Leaf	70 Images	30 Images
Anthracnose	70 Images	30 Images
Bird Eye Spot	70 Images	30 Images
Healthy	70 Images	30 Images

Fig. 10. Class sample image of the dataset

Recall is the success rate in retrieving information. Recall can be calculated using Eq. (5).

$$Recall = \frac{TP}{TP + FN} \times 100\% \tag{5}$$

3 Design and Experiment

The percentage of datasets in the training data, and testing data is 70% and 30%, respectively. The training data serves to create a learning model, and the testing data is a sample of data that has never been seen before and is used to evaluate the performance of the deep learning model. We collected Kaggle (https://www.kaggle.com/datasets, accessed on January 3, 2023). Table 2 is the data set for training and testing.

The data set consists of 400 images, with 280 images used as training and the remaining 120 images used as testing. The training stage aims to train the neural network to recognize the given image input. The output of network training is training accuracy and training model. In the testing phase, this model is used to recognize new data. The data set that we use can be seen in Fig. 10.

4 Results and Discussion

In this research starts from the image input process, the input data is processed in the convolution layer using max pooling, and ReLU. The results of the convolution process will be collected and processed flatten. After dataset is entered then some experiments were carried out on tea leaf disease data using 50, 100, 150 epochs. One epoch means

Table 3. The Accuracy by Epochs

Epoch	Train Accuracy	Time Train	Test Accuracy
Epoch 50	99.28%	2103 s	94.16%
Epoch 100	100%	4034 s	94.16%
Epoch 150	100%	6389 s	95%
Average	99.76%		

learns from the training data its entirety. In CNN, the iterative learning process aims to achieve convergence of weight values. The use of epoch depends on the amount of data used. This can result in varying levels of accuracy depending on the method used. The purpose of this research is the classification of tea leaf disease images using darknet-19 and epochs.

One epoch means learns from the training data its entirety. In CNN, the iterative learning process aims to achieve convergence of weight values. The use of epoch depends on the amount of data used. This can result in varying levels of accuracy depending on the method used. The purpose of this research is the classification of tea leaf disease images using darknet-19 and epochs.

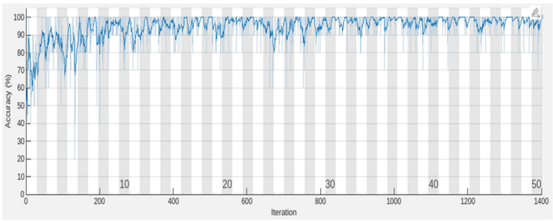
In this study, we used ADAM with a learning rate of 0.1. ADAM (Adaptive Moment Estimation) optimizer is used to train the data. The ADAM optimization technique will result in a 98% increase in accuracy [3]. ADAM functions to optimize a function and estimate moments. The accuracy of tea leaf disease classification can be seen in Table 3.

4.1 Comparison of Training and Testing Accuracy

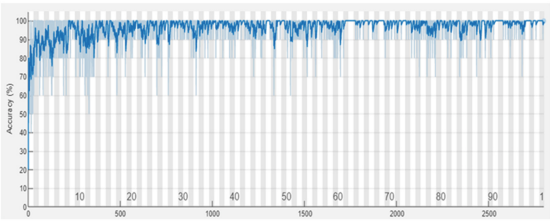
Epochs is intended to maximize the data to be processed and has been determined in mini batch. The use of epochs can improve accuracy. Epochs function as repetitions. Simply, the more epochs used, the better the accuracy obtained. In table III, it can be seen that the accuracy obtained increases as the epochs used increase. This is because the machine requires repetition on the dataset so that it to get maximum accuracy. The graphic in the training process is almost no significant difference, it can be seen in Fig. 11 and Fig. 12. And it can see in table iv that the average accuracy of each epoch is the same namely 25%. The accuracy value can illustrate good or not a classification model will be used to guess new objects or images. The average accuracy obtained is 99.76% in training data and 94.44% in testing data with details of the highest accuracy is at epoch 150.

4.2 Comparison of Computation Time

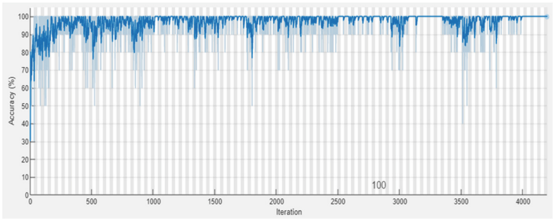
In this study, records were made of time required in processing image using the CNN algorithm. Based on Table 3, it can be seen that the higher the number of epochs, the longer the time required. This is because the more the number of epochs, the more repetitions that occur.



EPOCH 50



EPOCH 100



EPOCH 150

Fig. 11. Comparison of Plot Training on Three Epoch CNN.

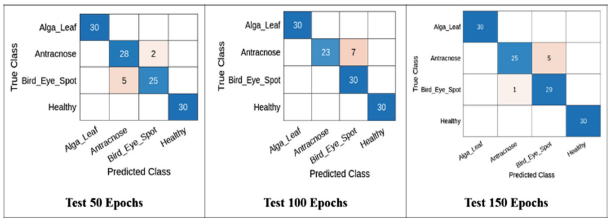


Fig. 12. Comparison Testing Data Results on Three Epochs.

Testing was conducted to determine the effect of the number of training (epochs) used. After the classification model is successfully created, the next step is to test to evaluate the model of the learning process results. The evaluation process uses the confusion matrix method. The parameters of this evaluation are accuracy, precision, and

recall. The result of the calculation of accuracy, precision and recall can be seen in the Table 4.

Table 4. Comparison by Epochs

EPOCH 50								AVERAGE
ALGAL LEAF		ANTRACNOSE		BIRD EYE		HEALTHY		
TP	30	TP	28	TP	25	TP	30	
FP	0	FP	2	FP	5	FP	0	
TN	0	TN	5	TN	2	TN	0	
FN	90	FN	85	FN	88	FN	90	
ACCURACY	25%		28%		23%		25%	25%
PRECISION	100%		93%		83%		100%	94%
RECALL	25%		25%		22%		25%	24%

EPOCH 100								AVERAGE
ALGAL LEAF		ANTRACNOSE		BIRD EYE		HEALTHY		
TP	30	TP	23	TP	30	TP	30	
FP	0	FP	7	FP	0	FP	0	
TN	0	TN	0	TN	7	TN	0	
FN	90	FN	90	FN	83	FN	90	
ACCURACY	25%		19%		31%		25%	25%
PRECISION	100%		77%		100%		100%	94%
RECALL	25%		20%		27%		25%	24%

EPOCH 150								AVERAGE
ALGAL LEAF		ANTRACNOSE		BIRD EYE		HEALTHY		
TP	30	TP	25	TP	29	TP	30	
FP	0	FP	5	FP	1	FP	0	
TN	0	TN	1	TN	5	TN	0	
FN	90	FN	89	FN	85	FN	90	
ACCURACY	25%		22%		28%		25%	25%
PRECISION	100%		83%		97%		100%	94%
RECALL	25%		22%		25%		25%	24%

4.3 Comparison of Testing Precision

Precision in the testing process is relatively the same, there is no noticeable difference because the number of datasets of each class is symmetrical or the same amount. The highest precision is at epoch 150 with value of 95%.

4.4 Comparison of Testing Recall

From table iv, it can be seen that the recall value in the testing process has a relatively similar value because the composition of the amount of data for each class has the same amount namely 30 data for each class.

5 Conclusions and Future Works

In our research we apply deep learning to classify the types of diseases that exist on tea leaves. As a comparison, we used three epochs are popular among researchers to see which model is more effective in our problem. Based on the results we obtained, it can be concluded that the best model was produced by epoch 150 with an accuracy value of 100% on training data and 95% on testing data, precision of 95% and recall of 24%. The use of epoch values also affects the accuracy of the model with dataset ratio of 70:30 with epochs of 50,100 and 150. The higher epoch, the better the resulting accuracy value. The difference in accuracy of each epoch is not too significant because each class has the same amount and has gone through the augmentation process. Of course, to make machines able to recognize types of diseases like humans, learning and testing is needed. The learning phase aims to train the machine, while the testing phase aims to measure the performance of the training model.

Acknowledgment. PUI-PT Combinatorics and Graphs, CGANT University of Jember, Mr. Alfian, and Mrs. Yuli who have provided assistance and motivated this research.

References

1. H.Gensheng, Y. Xiaowei, Z. Yan, W. Mingzhu "Identification of tea leaf diseases by using an improved deep convolutional neural network",*Journal Pre-proof,* 2019.
2. B.C. Karmokar, M.S. Ullah, Md.K.Siddiquee, and K. Md. Rokibul Alam, "Tea Leaf Recognition using Neural Network Ensemble", *International Journal of Computer Applications*, 2015.
3. D.K.Sharma, L.H. Son, R.Sharma, K.Cengiz, "Micro-electronics and telecommunication engineering", *Proceedings of* 4^{th} *ICMETE* , 2020.
4. S.Gayathri., D.C.Joy Winnie Wise, P. B.Shamini, N. Muthukumaran, "Image analysis and detection of tea leaf disease using deep learning", *Proceedings of the International Conference on Electronics and Sustainable Communication System*, 2020
5. M.Agarwal, A.Singh, S.Arjaria, A.Sinha, S.Gupta, "Tomato leaf disease detection using convolution neural network", *Procedia computer science,* 2020

6. R.Abduljabbar, H.Dia, S.Liyanage, S.A.Bagloee, “Applications of Artificial Intelligence in Transport : An Overview”, *Sustainability,* 2019
7. A.S.Rawat, A.Rana, A.Kumar, A.Bagwari, “Application of multi layer artificial neural network in the diagnosis system : A Systematic Review”, *International Journal of Artificial Intelligence*, 2018.
8. I.Kouretas, V.Paliouras “Simplified hardware implemetation of the softmax activation function”, *International Conference on Modern Circuits and Systems Technologies*, 2019.
9. R.S.Latha, G.R. Sreekanth, R.C. Suganthe, R.Rajadevi, S. Karthikeyan, S.Kanivel,B.Inbaraj, “Automatic detection of tea leaf diseases using deep convolution neural network”, *International Conference on Computer Communication and Informatics,* 2021.
10. M.Hashemi, “Enlarging smaller images before inputting into convolutional neural network : zero-padding vs interpolation”, *Journal Big Data*, 2019.
11. Y.Liu, H.Pu, D.W.Sun, “Efficient ectraction of deep image features using convolutional neural network (CNN) for applications in detecting and analysing complex food matrices”, *Trends in Food Science & Technology*, 2021.