Matrik: Jurnal Manajemen, Teknik Informatika, dan Rekayasa Komputer
Vol. 24, No. 2, March 2025, pp. 219∼234
ISSN: 2476-9843, accredited by Kemenristekdikti, Decree No: 200/M/KPT/2020
DOI: 10.30812/matrik.v24i1.4035

Identify the Condition of Corn Plants Using Gray Level Co-occurrence Matrix and Backpropagation

Abd Mizwar A. Rahim , Theopilus Bayu Sasongko
Universitas AMIKOM Yogyakarta, Indonesia

Article Info

Article history:

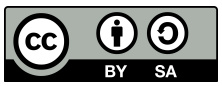
Received May 07, 2024
Revised February 14, 2025
Accepted March 06, 2025

Keywords:

Artificial Neural Networks;
Backpropagation;
Machine Learning;
Maize Leaf Disease;
Texture Matrix.

ABSTRACT

Maize production in Indonesia faces various challenges, including pest and disease attacks and dependence on rainfall, which impact yields. This research aims to develop an early detection system for corn leaf diseases to improve crop productivity. The methods used are the Gray Level Co-occurrence Matrix for feature extraction and Artificial Neural Network Backpropagation for classification. The dataset consists of images of corn leaves categorized as healthy, leaf-spot, leaf-blight, and leaf-rust. The research process includes data pre-processing such as normalization and background modification, feature extraction with Gray Level Co-occurrence Matrix, data balancing using Synthetic Minority Over-sampling Technique, and Artificial Neural Network Backpropagation model training. Evaluation was conducted using the Confusio% in the previous study. Applying pre-processing techniques increased the model's sensitivity to feature scaling, resulting in more accurate predictions than previous methods. This research improves the effectiveness of image-based plant disease classification. It opens up further research opportunities for parameter optimization in the Gray Level Co-occurrence Matrix and Backpropagation methods. The proposed system can be a solution for farmers to detect corn diseases quickly and accurately, thus supporting food security and agricultural efficiency.



Corresponding Author:

Abd Mizwar A. Rahim, +6285325973692,
Faculty of Computer Science and Informatics,
Universitas AMIKOM Yogyakarta, Indonesia,
Email: abdulmizwar@amikom.ac.id.

How to Cite:
A. A. Rahim and T. Sasongko, "Identify the Condition of Corn Plants Using Gray Level Co-occurrence Matrix and Bacpropagation", *MATRIK: Jurnal Manajemen, Teknik Informatika, dan Rekayasa Komputer*, Vol.24, No.2, pp. 219-234, March, 2025.

1. INTRODUCTION

Corn is one of the most important carbohydrate-producing food crops in the world. Apart from wheat and rice, corn seeds are a staple food for Central and South American people and some people in Africa and parts of Indonesia. Apart from that, corn is also an important component of animal feed [1, 2]. Corn production in Indonesia in 2020 was 29.02 million tons. The province with the largest national corn production was East Java, with a total production of 23.16 percent of national corn production in 2020 [3].

The opportunity to increase corn production to meet domestic and export needs is still quite large. The national corn improvement program to increase productivity and expand area will be implemented in different environments or agroecosystems, starting from high-productivity environments (optimal land) to low-productivity environments (marginal land/ dry); therefore, different and ecologically specific maize cultivation techniques are needed [4, 5]. One of the obstacles to producing corn is dry land, which is caused by a lack of water because it is very dependent on rainfall conditions. Apart from that, the level of pest and disease attacks is a factor that greatly influences the production of dry corn. The main causes of low corn yield in Indonesia are the use of local varieties, poor soil fertility, inadequate fertilization, and pest and disease attacks [6]. One way to find out if a corn plant is affected by disease is to look at the condition of the leaves on the corn plant. Corn plants indicated by disease usually will not bear fruit [7, 8]. Many farmers still need a long time to find out the condition of their corn plants; this can cause future corn plants not to grow according to the farmers' expectations [9].

Many ways can be done to help farmers quickly find out whether their corn plants are healthy or if they are infected with disease, one of which is by using Machine Learning, which has previously been proven to be able to solve topics like this [10, 11]. For example, classifying leaves that have medicinal properties. This research was carried out to inform people which leaves have medicinal properties and which do not. This research uses machine-learning methods [12].

Previous research generally did not normalize data and change the background, which can cause the model in carrying out classification not to get optimal results. This research aims to improve the accuracy results in identifying the condition of corn plants by proposing the gray level co-occurrence matrix method, backpropagation, and improving it by covering the shortcomings found in previous research to be able to improve accuracy results that are better than the accuracy produced previously, namely 98.4% carried out by Pratama P I, et al. [13].

Several previous studies have been conducted to classify corn leaf diseases using various artificial intelligence methods. Iswantoro & Handayani (2022) [14] applied the Convolutional Neural Network (CNN) method for corn leaf disease classification and achieved 94% accuracy. This research shows that the CNN algorithm is quite good at classifying images of corn leaves affected by disease. Furthermore, research conducted by Sibiya & Sumbwanyambe (2021) [15] used the Automatic Fuzzy Logic-Based Maize Common Rust Disease Severity Predictions approach with a combination of Thresholding and Deep Learning based on the VGG-16 network. This method produces 89% accuracy in identifying common rust disease in corn. Meanwhile, research by Kshyanaprava et al. (2020) [16] compared several machine learning algorithms in detecting corn leaf diseases. The results showed that the Random Forest (RF) method had the best performance with 79.23% accuracy, superior to other algorithms such as Naïve Bayes (NB), Decision Tree (DT), K-Nearest Neighbors (KNN), and Support Vector Machine (SVM). Another research was conducted by Pratama et al. (2022) [13] who also used the CNN method with the ResNet-50 and Adam Optimizer models. This study resulted in an accuracy rate of 98.4%, one of the highest achievements compared to previous studies. In addition, research by Nur A. Q. (2023) [17] applied AlexNet Convolutional Neural Network for corn leaf disease classification. This method's classification results reached 90% accuracy, showing that the AlexNet approach effectively detect diseases on corn leaves.

Of the various studies conducted, most use CNN methods and other machine-learning algorithms with varying degrees of accuracy. However, these studies still have some limitations, especially in data preprocessing aspects, such as normalization and background adjustment of corn leaf images. Therefore, this study offers a new approach combining the Gray Level Co-occurrence Matrix (GLCM) method for texture feature extraction and Artificial Neural Network (ANN) Backpropagation for classification. This approach is expected to increase the model's sensitivity to texture and feature scale variations, resulting in more accurate predictions than previous methods. This research hopes to develop a classification system that can detect corn diseases more accurately than previous research. Thus, this research contributes to the development of artificial intelligence-based agricultural technology, which can help farmers reduce the impact of plant diseases on crop yields while improving food security in Indonesia.

2. RESEARCH METHOD

The classification process for determining the condition of corn plants based on leaf images involves several steps using the GLCM method for feature extraction and ANN Backpropagation for classification. The process begins with dataset retrieval, where a dataset relevant to the topic is obtained from GitHub. The dataset consists of images converted into numerical texture features for classification. Next, during the data pre-processing stage, errors in the dataset are checked and corrected. The feature extraction

process then uses the GLCM method to calculate the appearance of the matrix in the pixels of each image, extracting numerical features. To address class imbalances in the dataset, the Synthetic Minority Over-sampling Technique (SMOTE) is applied in the data balancing stage. Following this, data normalization ensures uniformity in the data scale. The dataset is split into training and testing sets, with 70% used for training and 30% for testing. The classification process is carried out using the backpropagation method, identifying the condition of corn plants based on leaf features. The performance of the model is then assessed in the evaluation stage. If the accuracy does not surpass previous research results, adjustments are made to the parameters in the backpropagation process to achieve better accuracy. Once the accuracy exceeds prior research, the process moves to the final stage of concluding, where the results and findings of the research are summarized based on the designed model and testing outcomes. As shown in Figure 1, the research flow includes the entire process, from collecting the dataset to drawing conclusions.

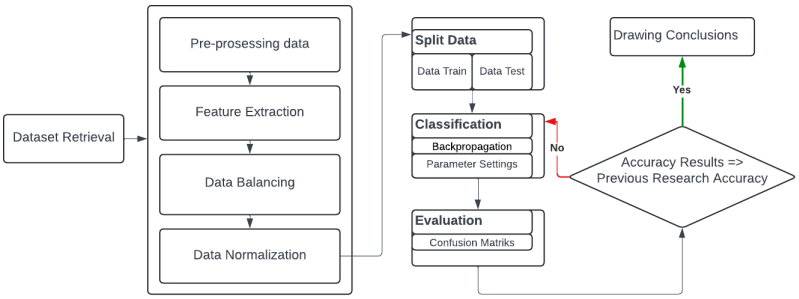


Figure 1. Research flow

Figure 1 Research Flow shows the systematic flow in this research, starting from the Dataset Retrieval stage, where a dataset of corn leaf images is collected for analysis purposes. Next, Data Pre-processing includes image normalization, color to grayscale conversion, and noise removal to improve data quality. After that, the Feature Extraction stage using the Gray Level Co-occurrence Matrix (GLCM) method is applied to extract texture features from the corn leaf images used in the classification process. To make the data distribution more balanced, Data Balancing is performed using the Synthetic Minority Over-sampling Technique (SMOTE) method, which aims to balance the amount of data in each class. Then, the extracted features are normalized in the Data Normalization stage to ensure a uniform data scale, thus improving the performance of the classification model.

After the normalization process, the dataset is divided into two parts in the Split Data stage: the Train Data used to train the model, and the Test Data used to test the model after the training process. The model is then classified using Artificial Neural Network (ANN) Backpropagation with certain parameter settings, which are tested to obtain the best accuracy. The Confusion Matrix evaluates the classification results, which displays the model's accuracy level in classifying corn leaf conditions. Suppose the accuracy obtained is higher or the same as the results of previous studies. In that case, the research continues to the Drawing Conclusions stage, where research conclusions are made based on the results of the model evaluation. However, suppose the accuracy obtained is still below the previous study. In that case, the model parameters will be adjusted, and the classification process will be repeated until more optimal accuracy is obtained. With this research flow, the proposed model is expected to improve the accuracy in automatically detecting corn leaf diseases, thus contributing to the development of artificial intelligence-based agricultural technology.

3. RESULT AND ANALYSIS

This section explains the results obtained from the calculation process. The research stages began with data retrieval, followed by preprocessing to ensure the dataset's quality and accuracy. Features were then extracted using the GLCM method to obtain texture values from the corn leaf images. Data balancing and normalization processes were carried out to address class imbalance and standardize the data for consistent analysis. Finally, classification was performed using backpropagation neural networks, and the model was evaluated using a confusion matrix to measure its performance.

3.1. Dataset

This research uses a corn leaf dataset; the dataset was taken from GitHub https://github.com/ibnujakaria/dataset-daun-jagung. This dataset is a dataset that contains a collection of images of diseased leaves and images of healthy leaves from corn plants. The leaf disease images in question are leaf spot, leaf blight, and leaf rust. This dataset consists of 3846 images of corn leaves and has

four classes: the leaf spot class with 508 images, leaf blight with 985 images, leaf rust with 1192 images, and healthy with 1162 images. This dataset is used to train and test classification models, especially in identifying the condition of corn plants based on leaf images. The following is a dataset display, which can be seen in Figure 2.

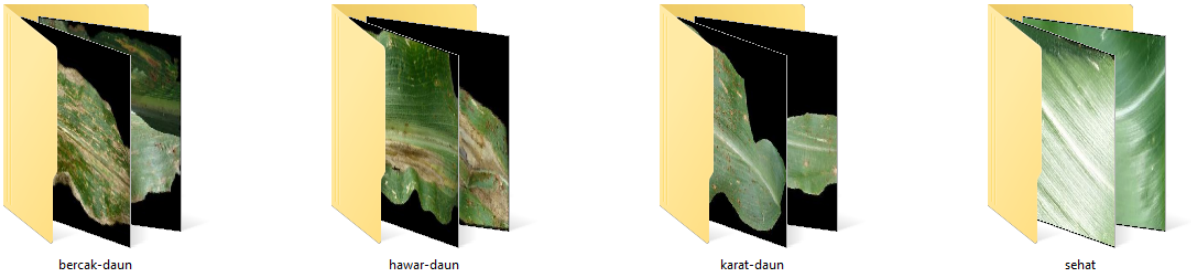


Figure 2. Dataset

In Figure 2 of the corn leaf dataset above, several classes are in it, the number of which experiences class imbalance because several classes in the target class have a larger number (majority class) compared to other classes in the target class that have a smaller number (majority class). Therefore, the Synthetic Minority Over-Sampling Technique (smote) method was applied to overcome datasets with class imbalance problems.

3.2. Data preprocessing

Before continuing with the feature extraction process, it is necessary to carry out a data pre-processing stage to get better feature values in the dataset [18, 19]. Several data pre-processing techniques exist: reducing image size (scaling), implementing grayscale, and hot coding. The results of the first technique, reducing the image dimensions, can be seen in Figure 3.

Figure 3. Scaling results

Figure 3 shows the process of reducing image dimensions/image resolution from the rust-leaf dataset class with initial dimensions of 513 × 579 pixels changed to 258 × 258 pixels. Each image from all classes is matched or equal in dimensions. This technique aims to reduce data errors that do not have an unbalanced value scale when the image is entered into the system. After the first technique is carried out, the process of changing the image to gray continues [20]. The results of this grayscale process can be seen in Figure 4.

Figure 4. Grayscalling process

Figure 4 above is the process of changing the image from RGB to a gray-level image. In one of the dataset classes, leaf rust,

which was originally green after the grayscale process, was changed to a gray pattern. This technique functions to remove the value of each image color and only leave the lighting value [21, 22]. This aims to eliminate variations in color values in the four types of leaves so that the system can read them easily and get better extraction calculation value results. In the context of an RGB image with 8-bit color depth per channel (8-bit/channel) for each color channel (R, G, and B), where the color values range from 0 to 255 (2^8 = 256 values), Converting an image to a grayscale image involves calculating a new pixel intensity value using the average formula of the R, G, and B values at that pixel. By applying the formula. Gray Value = (R G B)/3. The resulting gray value is still represented in 8 bits, ranging between 0 and 255. The next stage is the one-hot encoding process, namely changing the class to integer values 0 and 1. The results of the one hot encoding preprocessing technique can be seen in Table 1 below.

Table 1. One Hot Encoding Results.

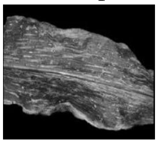
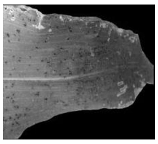
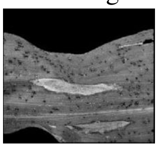
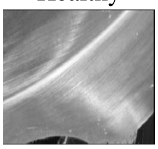
Class Label	**Feature Vector (One-Hot Encoding)**
Leaf-Spot 	1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
Leaf-rust 	0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
Leaf Blight 	0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0.0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0.0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
Healthy 	0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

Table 1 above shows the result of changing leaf type classes through the one-hot encoding process, where each leaf type has a different binary code number value. This one-hot encoding process is very effective on the system, making it easier for the system to detect the given number code. The four classes in this dataset exemplify the types of images.

3.3. Feature Extraction Using GLCM

In this study, manual calculation of the Gray Level Co-occurrence Matrix (GLCM) was carried out on the corn leaf image Figure 3 for the 0^o direction (horizontal to the right). The image was converted into grayscale format; then, the pixel intensity values were extracted to form a 4×4 GLCM matrix. This matrix is calculated based on the number of occurrences of pairs of pixel intensity values that are adjacent horizontally. The following are the results of the manual calculations that we have done: Step 1: Converting Images to Grayscale GLCM is usually calculated from grayscale images, so the first step is to convert this image to grayscale format. Definition of distance and direction: we will calculate GLCM based on the spatial relationship of pixels. Some common directions used: 0^o (Horizontal, right neighbor), 900^o (Vertical, bottom neighbor), 450^o (Bottom right diagonal), and 1350^o (Upper right diagonal); we will use the 00^o direction (horizontal, right neighbor) as an example; extract Pixel Values and Calculate GLCM; divide the image into a matrix of pixels with intensity values between 0 and 255; define the co-occurrence matrix, which

collates how often a pair of pixel intensity values occur in a particular direction. Normalize the matrix to get the probability of occurrence of a pixel pair. Now, we will manually calculate GLCM for this image. We will mathematically extract the key pixel values from the image and calculate the GLCM matrix. The given image will be converted into grayscale format. Grayscale means each pixel has an intensity value of 0 to 255. For example, after converting to grayscale and simplifying into a 4×4 matrix to make calculations easier, we get the following pixel value matrix.

$$I = \begin{bmatrix} 0 & 1 & 1 & 2 \\ 2 & 2 & 1 & 0 \\ 3 & 0 & 1 & 2 \\ 1 & 3 & 3 & 2 \end{bmatrix}$$

GLCM is created by counting how often each pair of pixels appears horizontally adjacent in the above matrix. We will record the occurrence of the pixel value pair (i, j), where: i is the current pixel value. j is the value of the neighboring pixel to its right. Next, Count Pixel Pairs; it is necessary to identify the pixel pairs (i, j) for the 0^o direction (horizontal, to the right): Row 1: (0,1), (1,1), (1,2), Row 2: (2,2), (2,1), (1,0), Row 3: (3,0), (0,1), (1,2), Row 4: (1,3), (3,3), (3,2) We count the number of occurrences of each pair, which can be seen in Table 2.

Table 2. GLCM Pixel Pair Occurrence Calculation Results

No	Pair (i,j)	Frequency
1	(0,1)	1
2	(1,1)	1
3	(1,2)	2
4	(2,2)	1
5	(2,1)	1
6	(1,0)	1
7	(3,0)	1
8	(0,1)	1
9	(1,2)	1
10	(1,3)	1
11	(3,3)	1
12	(3,2)	1

Next, we proceed to create the GLCM matrix based on the previously calculated pixel pairs. Since the pixel values in the image range from 0 to 3, we need to construct a 4×4 matrix to represent the co-occurrence of these values. The number 4 is chosen because there are four distinct gray levels present in the image. Each entry in this matrix represents the frequency of a specific pixel pair (i, j) appearing in a particular spatial relationship. The constructed GLCM matrix effectively captures the texture pattern of the image by analyzing pixel intensity relationships. The final result of the matrix formation is shown below, which will then be used for further texture feature extraction.

$$\begin{bmatrix} \frac{1}{j} & 0 & 1 & 2 & 3 \\ 0 & 0 & 2 & 0 & 0 \\ 1 & 1 & 1 & 3 & 1 \\ 2 & 1 & 1 & 1 & 0 \\ 3 & 1 & 0 & 1 & 2 \end{bmatrix}$$

This matrix represents the frequency of pixel value pairs (i, j) appearing in the image in the horizontal direction. To obtain the normalized GLCM, each value in the matrix is divided by the total number of occurrences of pixel pairs. The total number of pixel pair occurrences in this case is 12, which serves as the normalization factor. Normalization helps in standardizing the matrix values, making them easier to interpret and compare across different images. The resulting normalized GLCM matrix provides a probabilistic representation of pixel pair distributions. This matrix is then used in further calculations for extracting texture features such as contrast, energy, homogeneity, and correlation.

$$\begin{bmatrix} \frac{1}{j} & 0 & 1 & 2 & 3 \\ 0 & \frac{0}{12} & \frac{2}{12} & \frac{0}{12} & \frac{0}{12} \\ 1 & \frac{1}{12} & \frac{1}{12} & \frac{3}{12} & \frac{1}{12} \\ 2 & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} & \frac{0}{12} \\ 3 & \frac{1}{12} & \frac{0}{12} & \frac{1}{12} & \frac{2}{12} \end{bmatrix}$$

After dividing each value by 12, we obtain the final normalized GLCM matrix. This matrix represents the relative frequency of each pixel pair appearing in the image. By normalizing the values, the data becomes more interpretable and comparable across different images. Each element in the matrix now reflects the probability of a specific pixel pair occurring in the given spatial relationship. This normalized GLCM is essential for further texture feature calculations. It serves as the basis for extracting features such as contrast, energy, homogeneity, and correlation to characterize the texture properties of the image.

$$\begin{bmatrix} \frac{1}{j} & 0 & 1 & 2 & 3 \\ 0 & 0.00 & 0.17 & 0.00 & 0.00 \\ 1 & 0.08 & 0.08 & 0.25 & 0.08 \\ 2 & 0.08 & 0.08 & 0.08 & 0.00 \\ 3 & 0.08 & 0.00 & 0.08 & 0.17 \end{bmatrix}$$

This matrix represents the probability distribution of each pixel pair in the horizontal direction of the image. By normalizing the GLCM, we obtain a clearer representation of texture patterns based on pixel relationships. Following this step, the contrast texture feature is calculated to measure variations in pixel intensity within the image. Contrast highlights the degree of difference between adjacent pixel values, making it useful for identifying texture sharpness and edges. The computation of the contrast feature follows a predefined mathematical formula, as represented by Equation 1. This calculation is crucial in extracting meaningful texture information from the image.

$$Contrast = \sigma_{i,j} P(i,j)(i-j)^2 \tag{1}$$

Equation 1 is the formula of the contrast texture feature, one of the texture measure formulas in Gray-Level Co-occurrence Matrix (GLCM) analysis, used to evaluate pixel intensity variation in digital images. In this equation, P(i,j) is a joint probability matrix that indicates how often pairs of pixel values with intensities i and j appear in a given spatial pattern. The difference $(i-j)^2$ gives greater weight to larger intensity differences, so a high contrast value indicates the presence of sharp intensity differences in the image. This method is often used in image processing to detect edges, texture patterns, or surface roughness in various fields, such as object recognition, medical analysis, and satellite image processing. High contrast values indicate a rougher texture, while low values indicate a more homogeneous texture. After calculating the contrast, calculate the correlation to measure the relationship between the intensity values of adjacent pixels. Calculation of correlation texture features using equation 2.

$$Correlation = \frac{\Sigma i, j P(i,j)(i-\mu i)(j-\mu j)}{(\sigma i \sigma j)} \tag{2}$$

In equation 2, P(i,j) is the probability of occurrence of a pair of pixels with intensity values i and j in a particular spatial pattern. The parameters μi and μj represent the average value of pixel intensities in the row and column of the GLCM, respectively. At the same time, σi and σj are the standard deviations of the pixel intensities in the row and column. A high correlation value indicates a strong relationship between the pixel intensities in the image, which is often associated with structured or repetitive texture patterns. Conversely, a low correlation value indicates that the variation in pixel intensity is random, reflecting a coarser or irregular texture. Next, we can calculate Energy (Energy/Second Moment of Angle, ASM). Energy measures the uniformity of the matrix distribution. This texture feature calculation uses equation 3.

$$Energy = \Sigma_{i,j} P(i,j)^2 \tag{3}$$

In equation 3, P(i,j) is the probability of occurrence of a pair of pixels with intensities i and j in a particular spatial pattern. By summing the squares of these probability values, the energy indicates the extent to which the distribution of pixel intensity values in the image is uniform. Finally, we can calculate Homogeneity (Inverse Difference Moment, IDM) to measure how similar the intensities of adjacent pixels are. The formula of Homogeneity can be seen in equation 4.

$$Energy = \Sigma_{i,j} \frac{P(i,j)}{1 + |i - j|} \tag{4}$$

In equation 4, P(i,j) is the probability of occurrence of a pixel pair with intensities i and j, while $|i-j|$ is the absolute difference between their intensities. The divisor (1 + $|i - j|$) ensures that pixel pairs with similar intensities (closer i and j values) contribute more to the homogeneity value. If the Homogeneity value is high, the intensity differences in the image are small, indicating a smoother or uniform texture. Conversely, if this value is low, then there is a lot of variation in pixel intensities, reflecting a coarser or irregular texture. The results of the GLCM (0^o) calculations and their texture features can be seen in Table 3.

Table 3. Results of Manual Calculation of GLCM (0°) and its Texture Features

No	Feature	Value
1	Contrast	2.02
2	Correlation	0.89
3	Energy (ASM)	0.1651
4	Homogeneity	0.79

The dataset extraction results are saved in a .csv file format. This process is executed using a Python program for data processing. The feature extraction results are generated from the dataset images after applying the GLCM method. These results are stored for further analysis and classification. The overall feature extraction results are visualized in Table 4 for better understanding and reference.

Table 4. Gray Level Co-occurrence Matrix (GLCM) Extraction Results

dissimilarity_ 0	dissimilarity_ 45	dissimilarity_ 90	dissimilarity_ 135	correlation_ 0	correlation_ 45	correlation_ 90	correlation_ 135	homogeneity_ 0
24.0165239	24.7100592	16.74541	22.7061144	0.63770123	0.65810308	0.8482546	0.67705768	0.07021087
21.2735618	28.6423406	29.8317014	30.2998028	0.45441925	0.1461921	0.08547139	0.05566733	0.06521428
15.4192167	18.1176857	19.002448	20.0539119	0.51559968	0.29611867	0.27570816	0.20431487	0.06711598
13.0832313	38.3872452	39.8837209	37.7968442	0.80702288	-0.209875	-0.2205806	-0.172067	0.13051329
22.255814	30.035503	29.5966952	28.678501	0.50953281	0.09362214	0.09431012	0.16027136	0.05329854
14.2809058	25.8021039	29.1450428	28.8869165	0.7652593	0.18029117	0.03222921	0.0211572	0.07286776
24.0813954	54.4970414	54.9791922	49.6403682	0.76756786	-0.1430252	-0.1282736	-0.0082192	0.05447857
20.6493268	38.8224852	37.8108935	38.3155819	0.73666652	0.21042006	0.23782921	0.24072656	0.05706638
13.5507956	24.3984221	24.873929	24.3846154	0.65784446	-0.0995244	-0.0929249	-0.0802345	0.08235455
22.1119951	50.0407627	57.1872705	52.321499	0.88539814	0.43148444	0.29024978	0.3362021	0.06559597
17.380661	30.6252466	32.5244798	31.9119001	0.76077634	0.25449275	0.17850017	0.18255564	0.06989902
20.2399021	15.0907298	19.9761322	22.6857331	0.40897782	0.73519124	0.52407761	0.28106839	0.07943338
8.22337821	9.75542406	9.20563036	9.58842867	0.34635599	0.11459644	0.22454712	0.17038348	0.13451794
12.9932681	23.9822485	24.628519	22.6120973	0.8113019	0.29303915	0.30100161	0.34344063	0.09148164
18.7723378	34.3708087	34.0679315	25.817883	0.67047361	0.11683454	0.13615038	0.43548478	0.12257846
20.2086903	21.9119001	17.381273	20.103879	0.64694286	0.56840034	0.68521206	0.59638442	0.0554305
22.1768666	42.2649573	39.7068544	34.3168968	0.82181196	0.42828228	0.49490839	0.63406428	0.05644985
16.2099143	24.34714	29.25459	28.4852071	0.71619149	0.27024152	0.06299963	0.10795642	0.07410553
13.8959608	19.6844182	19.7845777	18.9349112	0.63572187	0.38864984	0.43517611	0.44441497	0.08517859
19.25459	17.9506903	20.0434517	20.2439185	0.14003686	0.25646623	0.09105162	0.0654084	0.05159684

3.4. Balancing Dataset

To overcome the imbalance that occurs, this research uses the Synthetic Minority Over-Sampling Technique (SMOTE) method to overcome this by adding a minority class so that it is the same as the majority class by adding artificial data, artificial or synthetic data created based on k-nearest neighbors. SMOTE selects k nearest neighbor samples from the minority class and generates new samples among them using formula 5.

$$Xnew = Xi + \lambda \times (Xk - Xi) \tag{5}$$

In the data synthesis process using the SMOTE method presented in equation 5, Xnew is a new synthetic data sample generated to balance the class distribution. This sample is formed based on Xi, a minority data point selected from the original dataset for reference when creating new data. Furthermore, one of Xi's nearest neighbors, referred to as Xk, is used in the interpolation process. To generate Xnew, this method utilizes a random value λ between 0 and 1, thus allowing the creation of new samples between Xi and Xk proportionally. In this way, SMOTE can expand the representation of the minority class more variedly without simply duplicating existing data directly. For an example of initial data, suppose we have a dataset with two features for the minority class in Table 5.

Table 5. Example of Initial Data

Sample	Feature 1 (X_1)	Feature 1 (X_2)
A	2.0	3.0
B	2.2	3.5
C	2.4	3.2

Table 5 Our goal is to add one synthetic sample to increase the number of minority data; select a Random Point from the Minority Class; at this stage, we choose one of the data points from the minority class as a reference for generating synthetic samples. Point A (2.0, 3.0) is selected as Xi in this example. Next, we determine the nearest neighbor to this point using Euclidean distance. Calculating the distance between points A and B produces a value of about 0.54, while the distance between A and C is 0.45. Since point C (2.4, 3.2) has a smaller distance to A, point C is selected as the nearest neighbor Xk to be used for interpolation in generating synthetic samples.

After determining the pair of points A (2.0, 3.0) and C (2.4, 3.2), the next step is to generate new synthetic samples using linear interpolation. Taking a random value of λ = 0.6, the SMOTE formula calculates the position of a new point between A and C. This process is done by adding 60% of the difference in coordinate values between A and C to point A, which produces a new sample at (2.24, 3.12). This sample is then added to the dataset as additional synthetic data, thereby increasing the number of minority class samples without simply duplicating the existing data. The results of the newly added synthetic data can be seen in Table 6.

Table 6. Manual Results of Smooth Calculation

Sample	Feature 1 (X_1)	Feature 1 (X_2)
A	2.0	3.0
B	2.2	3.5
C	2.4	3.2
New	**2.24**	**3.12**

The dataset presented in Table 7 illustrates the results of class balancing after adding 921 new data points. Various statistical features, including dissimilarity, correlation, and homogeneity across angles, are displayed for the newly balanced dataset. These features provide insights into the data's structural and textural characteristics. The addition of synthetic data ensures a more even distribution of classes, improving the model's performance in classification tasks. A balanced dataset improves the reliability of subsequent analyses by reducing bias toward the majority class.

Table 7. Dataset Class Balancing Results

No	dissimilarity_ 45	dissimilarity_ 90	dissimilarity_ 135	correlation_ 0	correlation_ 45	correlation_ 90	correlation_ 135	homogeneity_ 0
4747	3513.81332	8.12384693	9.19782516	9.11070334	10.1766092	0.76331499	0.69058572	0.70195071
4748	3323.27377	8.57165085	8.90157455	7.35163034	7.80631585	0.70541407	0.67356875	0.77739751
4749	3032.80495	14.3119357	13.4295105	21.919271	24.1001297	0.65277295	0.72200271	0.22276804
4750	3209.67142	12.5448454	15.8630228	16.4883616	15.9859122	0.72737434	0.56805309	0.5460575
4751	3115.38875	17.3295147	18.9649562	17.4829428	10.8908868	0.55997719	0.4141081	0.54985051
4752	3492.52171	14.8255061	30.2464244	30.7163037	29.2996405	0.85654542	0.37927179	0.38710599
4753	2926.76874	4.31356148	7.03828269	6.99435355	6.96527567	0.68115796	0.218499	0.2044138
4754	3314.63956	8.02352078	19.1214027	22.782378	21.1412535	0.9173217	0.52492619	0.36859472

Table 7 (*continued*)

No	dissimilarity_ 45	dissimilarity_ 90	dissimilarity_ 135	correlation_ 0	correlation_ 45	correlation_ 90	correlation_ 135	homogeneity_ 0
4756	3700.93	14.5886808	38.591554	40.0588957	33.998979	0.87108068	0.16716179	0.12052874
4757	2928.59542	21.2200922	18.475543	32.1910913	34.7558259	0.68261395	0.78499654	0.30542001
4758	2849.57466	11.4124111	7.37743804	12.5807106	14.8521149	0.65534998	0.90278603	0.53302745
4759	3092.60842	13.9932274	8.96036628	19.4858478	23.1222008	0.75550281	0.90609691	0.54021151
4760	2959.03385	10.9332157	15.1752466	18.414449	18.3067508	0.77724585	0.54829641	0.40943854
4761	3697.45964	15.3057614	13.5233458	20.0255446	23.2787344	0.75187351	0.77318205	0.56530577
4747	3513.81332	8.12384693	9.19782516	9.11070334	10.1766092	0.76331499	0.69058572	0.70195071
4748	3323.27377	8.57165085	8.90157455	7.35163034	7.80631585	0.70541407	0.67356875	0.77739751
4749	3032.80495	14.3119357	13.4295105	21.919271	24.1001297	0.65277295	0.72200271	0.22276804
4750	3209.67142	12.5448454	15.8630228	16.4883616	15.9859122	0.72737434	0.56805309	0.5460575
4751	3115.38875	17.3295147	18.9649562	17.4829428	10.8908868	0.55997719	0.4141081	0.54985051

In Figure 7 above, some data has been balanced using smote on the corn leaf dataset. The results are the values of the dataset features in the form of synthetic data built by Smote. Figure 6 displays the results of balancing 20 data sets, starting from the data from smote 4747 and moving up to the data -4766. The number of classes added is the minority class, including the leaf spot class, which was originally 508; 684 synthetic data were added so that the number of leaf spot classes became 1192, next for the minority class, namely the leaf blight class, which was originally 985, 207 synthetic data were added so that the total from the leaf-blight class to 1192, and to the healthy-leaf class which was originally 1162, 30 synthetic data were added so that the number of healthy-leaf classes became 1192.

3.5. Data normalization

This study applies data normalization to equalize the feature scale so that each attribute has a uniform value range, which can improve the stability and accuracy of the model in the classification process. Normalization in the GLCM (Gray Level Co-occurrence Matrix) method ensures that each value in the matrix has a probability scale of 1 so that texture features such as contrast, correlation, energy, and homogeneity can be calculated more accurately. In addition, normalization also reduces the impact of scale differences between features with large and small values so that the model can be more sensitive to texture patterns in corn leaf images.

This research dataset has a different scale for each attribute, for example, the value contrast_145 and the ASM_0 column, which can be seen in Figure 5. GLCM Extraction Results, so it is necessary to standardize it to have the same scale when building a machine learning model. The data normalization technique used is Min Max normalization, a normalization method that involves a linear transformation of the original data to produce a balance of comparison values between attributes. When these attributes are converted and produce similarity calculations, they can then be in the range 0 to 1. The following are the standardization results using the min_max normalization technique on the stroke prediction dataset, which can be seen in Figure 5.

```
1  from sklearn.preprocessing import MinMaxScaler

1  Scaler = MinMaxScaler()

1  Scaler.fit_transform(Xtrain,Ytrain)

array([[0.20906754, 0.19880725, 0.20749111, ..., 0.00347713, 0.00333337,
        0.0024156 ],
       [0.25029795, 0.1860415 , 0.21767031, ..., 0.00404141, 0.00316655,
        0.00263158],
       [0.22395028, 0.13756517, 0.19372118, ..., 0.00468798, 0.00496238,
        0.00330694],
       ...,
       [0.25986191, 0.2473714 , 0.22043748, ..., 0.00406823, 0.00377776,
        0.00416237],
       [0.19152541, 0.18676002, 0.21702464, ..., 0.00597076, 0.00489139,
        0.00483004],
       [0.22527013, 0.3721309 , 0.28142707, ..., 0.00750102, 0.00756131,
        0.01167479]])
```

Figure 5. Normalization results

3.6. Split data

Dividing the dataset into training data and testing data, the division in this research divided training data and testing data into 70/30. With this number of divisions, the aim is to see the model in predicting when it has test data with a total of 1431. In general, machine learning models get good accuracy results if they have a small amount of testing data. So, in this research, we increase the test data and test whether the model gets good results or not. Table 8 describes the data distribution that was carried out.

Table 8. Train/Test Split

Information	Training Data	Testing Data
Proportion	70%	30%
Amount	3337	1431

3.7. Backpropagation Classification

Backpropagation is calculated in two main stages: forward propagation to calculate the output and backward propagation to update the weights based on the prediction error. In the forward propagation stage, the input data X1 and X2 are multiplied by the initial weights W1 and W2 and then added to the bias b. This result is calculated as the initial activation value on the hidden neuron using equation 6.

$$Z = (X1 \times W1) + (X2 \times W2) + b \tag{6}$$

In equation 6, X_1 and X_2 represent the input feature values used in the classification process, while W_1 and W_2 are the weights calibrated during model training. Bias b serves as an additional value that helps shift the activation results so that the model is more flexible in adjusting to data patterns. The calculated result of Z is then used as input for the activation function that determines the final output of the neuron. This process is carried out iteratively in the artificial neural network, with weight updates through backpropagation to optimize the model's performance in the classification task. The Z value is then passed through a sigmoid activation function formulated as equation 7.

$$A = \frac{A}{1 + e^{-z}} \tag{7}$$

In equation 7, Z results from a linear combination of inputs, weights, and biases before being applied to the activation function. The sigmoid function maps the Z values into a range between 0 and 1, making it suitable for probabilities in classification problems; when Z is very large, A approaches 1, while when Z is very small (large negative), A approaches 0. The results of the hidden layer activation are then multiplied by the output weight of the W3 layer, and bias is added before recalculating using the sigmoid activation function to produce the predicted value $Y^{\wedge}$. After obtaining the predicted results, the backward propagation stage is carried out by calculating the error using the Mean Squared Error (MSE) function in equation 8.

$$J = \frac{1}{2}(Y - Y^{\wedge})^2 \tag{8}$$

In equation 8, the difference between Y and $\hat{Y}$ is squared to ensure that the error is always positive and to emphasize larger errors more. The factor 1/2 is used to facilitate the calculation of derivatives in the weight update process during backpropagation. This loss function helps the neural network adjust the weights so that the predicted value is closer to the actual value. Thus, the smaller the value of J, the better the model's performance in predicting the expected output. Next, the derivative of the loss function is calculated, based on the weights, to obtain the error gradient using equation 9.

$$\partial J = \frac{\partial J}{\partial A} \times \frac{\partial A}{\partial Z} \times \frac{\partial Z}{\partial w} \tag{9}$$

In equation 9, $\frac{\partial J}{\partial A}$ represents the change of the loss function (J) to the activation output (A), which shows how much the activation contributes to the model error. Furthermore, $\frac{\partial A}{\partial Z}$ is the derivative of the activation function to the input value before activation (Z), which, in the case of the sigmoid function, will give a value that depends on the sigmoid output itself. Meanwhile, $\frac{\partial Z}{\partial W}$ represents the change of Z to the weight (W), which shows how the weight affects the output of the neuron. Using these chained

derivatives, the neural network weights can be updated iteratively through the gradient descent method to minimize the error in the learning process. Since the activation function is sigmoid, its derivative is calculated using equation 10.

$$\frac{d}{dx}f(x) = A(1 - A) \tag{10}$$

Equation 10 is the derivative of the sigmoid activation function, which is widely used in artificial neural networks for classification tasks. In this context, A is the output of the sigmoid function given by $A = 1/(1 + e\hat{}(-Z))$. This derivative shows how small changes in x (input value) affect the output A. The unique property of this sigmoid derivative is that its values always lie in the range 0 to 1, which helps in controlling the weight updates during backpropagation. In addition, the form A(1 - A) ensures that the gradient never becomes zero for values of A between 0 and 1, thus remaining effective in the neural network's learning process. After calculating the gradient, the weights are updated using Gradient Descent with equation 11.

$$W_{BARU} = W_{LAMA} - \alpha \times \frac{\partial J}{\partial W} \tag{11}$$

Equation 11 is the weight update formula in the Gradient Descent algorithm used in the backpropagation process in artificial neural networks. In this equation, W_{Old} is the weight value before being updated, while W_{New} is the weight value updated after a learning iteration. The parameter α (alpha) represents the learning rate, a scalar factor controlling each iteration's weight update step's size. The gradient $\frac{\partial J}{\partial W}$ indicates the change in the value of the loss function (J) to the weight (W), which is used to direct the optimization process so that the weight moves in a direction that reduces the error. With this approach, the neural network gradually adjusts its weights to reach an optimal solution for more accurate predictions. Tests using the backpropagation method were carried out to determine the ability of a model to identify four types of diseases in corn plants based on leaf images. The system's ability to identify leaf types depends on the backpropagation training process to accurately identify the leaf types used in the testing stage. The backpropagation parameters used in the training stage can be seen in Table 9.

Table 9. Backpropagation Parameter Settings

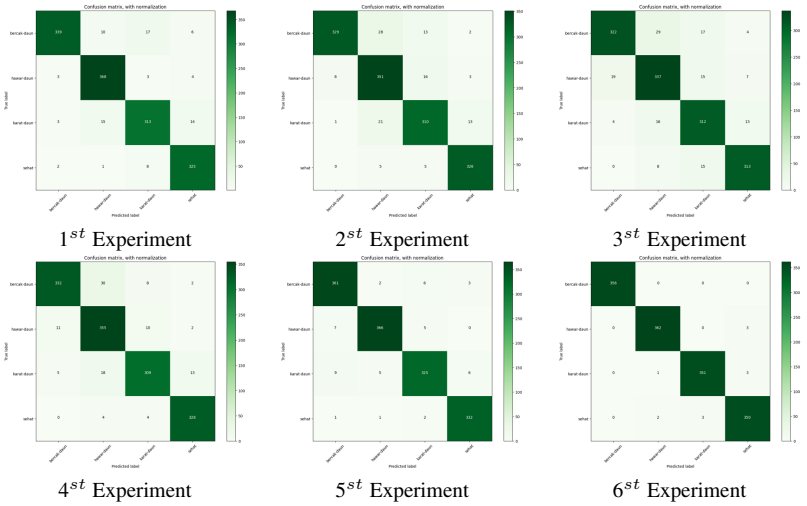
No	Input Neuron	Hidden Neuron	Output Neuron	Epoch	Batch Size	Accuracy	Early Stopping
1	24	50, 100, 150, 200, 250	4	1000	42	94%	Epoch 13 : Early Stopping
2	24	50, 100, 150, 200, 250	4	2000	42	92%	Epoch 16 : Early Stopping
3	24	40, 80, 125, 150, 220	4	1000	42	90%	Epoch 9 : Early Stopping
4	24	25, 75, 125, 145, 215	4	1000	42	93%	Epoch 12 : Early Stopping
5	24	32, 50, 100, 150, 200, 250,70	4	1000	22	97%	Epoch 12 : Early Stopping
6	24	32, 50, 100, 150, 200, 250	4	1000	32	99%	Epoch 6 : Early Stopping

In Table 9, the parameters used are input neuron, hidden neuron, output neuron, epoch, and batch size. The hidden layer uses ELU (Exponential Linear Unit) activation, and the last hidden layer uses Sigmoid activation. The split data used is 70% for training data and 30% for testing data; the number of trials is six classification trials. Each of these experiments has a different set of parameters. The stages of this research consist of 7 stages. The stages carried out are data preprocessing, feature extraction using GLCM, data balancing, data normalization, data split, classification using backpropagation, and evaluation using a confusion matrix. Of the 6 experiments, the sixth experiment obtained the best accuracy results, namely 99%, by having 24 input neurons. The number of input neurons was 24 because it equates to the number of features in the dataset totaling 24 columns. This input neuron will bring data into the system to then continue with the next layer process; there are six hidden layers with values of 32, 50, 100, 150, 200, and 250 neurons in each layer; the meaning of the hidden layer value is the randomization value of each experiment carried out, from randomizing neuron increments and neuron decrements to be sent to epoch as one of the stages to obtain accuracy. The output layer consists of 4 neurons that are the same as the dimensions of 4 categorical columns in y_train and y_test; this equates to the number of classes in the dataset, which is four classes. The number of epochs and batch sizes that are available is 1000 and 32, which means that epochs are the number of complete passes that must be carried out on the training dataset that has been determined because, in previous experiments, lowering or increasing epochs did not produce better accuracy. For batch size 32, several samples were processed before the model was updated, where the batch size value of 32 was also set because previous experiments using batch sizes of more than or less than 32 produced no better accuracy. In Table 9 above , which consists of 6 experiments, the first experiment and the second experiment have the same parameter values except for the number of epochs, the first experiment has an epoch of 1000 and the second experiment has an epoch of 2000, the second experiment is not better than the first experiment, the accuracy of the second experiment The result obtained was 92%, while the first experiment obtained an accuracy of 94%. Then, in

the third and fourth experiments with the same parameters except for the value of the hidden layer, which was lower than the 1st and 2nd experiments, the accuracy was still lower than the first experiment. The accuracy obtained was 90% for the third experiment and 93% for the third experiment. fourth. Of the four experiments, the best results are from the first experiment. for the 5th experiment, which had the addition of two hidden layers so, there were a total of 7 hidden layers and had a different batch size value of 22, and the other parameters had the same values as the previous highest experiment; the results obtained had the best accuracy, namely 97%. In the last experiment, only one hidden layer was added, so the number of hidden layers was 6. Other parameters besides the hidden layer had the same values as the parameter values in the previous experiment, and the accuracy obtained was 99%. Another technique used in each of these tests is the Early stopping technique. This technique involves monitoring model performance on validation data during training. Training is stopped early if performance on validation data no longer improves after a few iterations. This process is carried out to make it faster and more efficient to determine which iteration has the best results. The test results show that the default Configuration for EarlyStopping in Keras includes several key parameters that influence the behavior of the early-stopping technique during model training. First, the monitor='val_loss' parameter indicates that the metric to be monitored is the loss value in the validation data. Furthermore, with patience=0, training will stop as soon as deterioration occurs in the monitored metric without waiting for additional epochs. Setting verbose=0 indicates that messages during training will not be displayed in detail. The 'auto' mode in the mode parameters adjusts the monitoring direction to stop early, with training stopping when the monitored metric stops increasing. The parameter baseline=None allows comparing monitored metrics against certain reference values; training will be stopped early if metrics do not improve or decrease from baseline values. Finally, with restore_best_weights=False, the model weights will not be restored to the best weights achieved during training when early stopping is applied. Overall, this configuration provides flexible control to adjust early stopping behavior based on evaluating model performance on validation data. In the 1st, 4th, and 5th tests, training was stopped early after reaching the 13th or 12th epoch, indicating that the model performance did not experience a significant increase in the loss value on the validation data after passing the patience limit. Meanwhile, in the 2nd test, the model was stopped at the 16th epoch, and in the 3rd and 6th tests, training ended early at the 9th and 6th epoch, showing variations in the model's response to the early configuration. stopping. These results provide an overview of the impact of model sensitivity on training parameters and the effectiveness of using early stopping to optimize the training process by avoiding overfitting to validation data.

3.8. Evaluation of the method using Confusion Matrix and Classification Report

Of the six experiments carried out to identify corn plant diseases or no disease indications using a backpropagation artificial neural network, as is known, the experiment that produced the best accuracy was in the sixth experiment using the best input layer, hidden layer, an output layer, epoch, batch size values. The confusion matrix results will produce a display of correct predictions and incorrect predictions identifying disease in corn plants or no indication of disease in tabular form. The sixth experiment is the best result of all the previous classification processes. The confusion matrix is displayed. The results of the confusion matrix from all the experiments carried out can be seen in Figure 4.



1^{st} Experiment 2^{st} Experiment 3^{st} Experiment

4^{st} Experiment 5^{st} Experiment 6^{st} Experiment

Figure 6. Confusion matrix results

Figure 6 shows the overall results of the confusion matrix in the experiments carried out, and the best results were in the 6th experiment. The following is an explanation of the results of the confusion matrix in the 6th experiment: One category is predicted correctly as a whole, namely the type of leaf spot image, with a total of 356 test data. A total of 3 data categories for leaf blight, four categories for leaf rust, and five categories for healthy were identified incorrectly, namely predicted as three healthy, 1 for leaf blight, 3 for leaf rust, and 2 for leaf blight. From the results of testing the identification of types of corn leaf disease based on leaf images using backpropagation artificial neural networks, it can be seen that the best results from the experiments carried out with several parameters were tested, namely in the sixth experiment, the best accuracy results were 99% which were displayed through the classification report in the confusion matrix section. can be seen in Figure 7.

	precision	recall	f1-score	support
bercak-daun	1.00	1.00	1.00	356
hawar-daun	0.99	0.99	0.99	365
karat-daun	0.99	0.99	0.99	355
sehat	0.98	0.99	0.98	355
accuracy			0.99	1431
macro avg	0.99	0.99	0.99	1431
weighted avg	0.99	0.99	0.99	1431

Figure 7. Classification report results of the sixth experiment

Figure 7 shows the results of the Classification Report on the best experiment. The accuracy results from this experiment produced an accuracy of 99%, with this accuracy being the ratio of correct predictions in identifying corn plant diseases based on leaf images. This study proposes a method for identifying corn plant conditions using a combination of Gray Level Co-occurrence Matrix (GLCM) for feature extraction and Artificial Neural Network (ANN) Backpropagation for classification. The research process begins with dataset retrieval, which then goes through a preprocessing stage, such as normalization and image background modification, to improve data quality. After that, texture features are extracted using GLCM, which produces numerical values based on corn leaf texture patterns. To overcome the class imbalance in the dataset, the Synthetic Minority Over-sampling Technique (SMOTE) method is applied by adding synthetic samples to the minority class. After the data normalization process to equalize the feature scale, the dataset is divided into 70% training data and 30% test data before being classified using a Backpropagation Neural Network. The learning process is carried out by adjusting the network weights using Gradient Descent, updated gradually based on the calculation of Mean Squared Error (MSE) until the model reaches convergence.

Model evaluation using a confusion matrix shows that the developed method has achieved 99% accuracy, higher than previous studies, which obtained 98%. These results confirm that the use of preprocessing techniques such as normalization and background modification of the dataset significantly improves the performance of machine learning models. In addition, integrating GLCM with Backpropagation proved more effective than previous approaches in classifying corn leaf diseases. Thus, this study provides an important contribution to the field of artificial intelligence-based plant disease detection, as well as opening up opportunities for further exploration of GLCM parameter optimization, ANN architecture, and the use of deep learning methods to improve the accuracy and efficiency of plant disease identification systems.

4. CONCLUSION

Based on the results of this research analysis, it can be concluded that this study has shown that the combination of the Gray Level Co-occurrence Matrix (GLCM) method for feature extraction and Artificial Neural Network (ANN) Backpropagation for classification can improve the accuracy of corn leaf disease identification. Applying data preprocessing, normalization, and class balancing techniques using SMOTE has improved model performance in distinguishing healthy and diseased leaf texture patterns. These results indicate that a machine learning-based approach can effectively support early detection of plant diseases automatically. This study implies that an artificial intelligence-based classification system can be applied in a smart farming system to help farmers identify diseases faster and more accurately, thereby reducing the risk of crop failure. In addition, the developed method can be adapted for disease analysis in other plants with similar texture patterns, making it flexible for various applications in the agricultural field. However, this study has several limitations, including the limited number of datasets, which can affect the model's generalization when applied to more complex field conditions. In addition, this study only uses GLCM as a feature extraction method. In contrast, other methods, such as Wavelet Transform or deep learning-based feature extraction, can be explored to improve prediction

accuracy. For future research, developing a more robust classification system can be done by integrating deep learning architectures such as Convolutional Neural Network (CNN) or Hybrid Models that combine GLCM-based feature extraction techniques and deep learning methods. In addition, implementing the system as a mobile application or cloud-based platform can provide wider benefits to the community, especially for farmers who need practical solutions for detecting plant diseases. Thus, this study contributes to the development of artificial intelligence-based agricultural technology, supports food security, and opens up opportunities for further innovation in agricultural image classification.

5. ACKNOWLEDGEMENTS

Thank you for the financial assistance and support of this research from the Institute for Research and Community Service, Amikom University Yogyakarta.

6. DECLARATIONS

AUTHOR CONTIBUTION

Author 1 is responsible for dataset acquisition, pre-processing, feature extraction, balancing, normalization, and division into training and testing data. We thank Author 2 for making valuable contributions by implementing the classification method using Backpropagation, setting parameters, and conducting model evaluation. We also thank Author 2 for carefully analyzing the results and providing valuable insights into the conclusion.

FUNDING STATEMENT

In an internal grant scheme, this research was supported by funding from Universitas Amikom Yogyakarta through the Institute for Research and Community Service (LPPM). No external sponsorship influenced the research design, data analysis, interpretation of results, or writing of this article.

COMPETING INTEREST

The authors declare that no conflicts of interest are associated with this research. No financial, personal, or professional interests could influence the results or interpretation of this research.

REFERENCES

[1] F. Rozi, A. B. Santoso, I. G. A. P. Mahendri, R. T. P. Hutapea, D. Wamaer, V. Siagian, D. A. A. Elisabeth, S. Sugiono, H. Handoko, H. Subagio, and A. Syam, "Indonesian market demand patterns for food commodity sources of carbohydrates in facing the global food crisis," *Heliyon*, vol. 9, no. 6, p. e16809, 2023, https://doi.org/10.1016/j.heliyon.2023.e16809.

[2] 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, 2021, https://doi.org/10.15294/biosaintifika.v13i3.30527.

[3] S. Sugeng and A. Fitria, "Food Sovereignty For Indonesia: The Epistemological Dimension of Knowledge and Variety of Local Food," vol. 6, no. 1, pp. 18–32, 2023, https://doi.org/10.38043/jah.v6i1.4179.

[4] E. Jones-Garcia and V. V. Krishna, "Farmer adoption of sustainable intensification technologies in the maize systems of the Global South. A review," *Agronomy for Sustainable Development*, vol. 41, no. 1, pp. 8–16, 2021, https://doi.org/10.1007/s13593-020-00658-9.

[5] P. Hajong, S. Mondal, M. A. Islam, and A. Ghosh, "Economics of maize cultivation at selected intensive areas of Bangladesh," *International Journal of Agricultural Research, Innovation and Technology*, vol. 13, no. 2, pp. 70–78, 2024, https://doi.org/10.3329/ijarit.v13i2.70859.

[6] A. Wahyuni, J. Sujono, and Istiarto, "Evaluation of irrigation channels in the dry season at Dadahup swamp irrigation area to strengthen food security," *IOP Conf Ser Earth Environ Sci*, vol. 1311, no. 1, p. 012046, 2024, https://doi.org/10.1088/1755-1315/1311/1/012046.

[7] R. Rashid, W. Aslam, R. Aziz, and G. Aldehim, "An Early and Smart Detection of Corn Plant Leaf Diseases Using IoT and Deep Learning Multi-Models," *IEEE Access*, vol. 12, no. 2, pp. 23 149–23 162, 2024, https://doi.org/10.1109/ACCESS.2024.3357099.

[8] D. S. Joseph, P. M. Pawar, and K. Chakradeo, "Real-Time Plant Disease Dataset Development and Detection of Plant Disease Using Deep Learning," *IEEE Access*, vol. 12, pp. 16 310–16 333, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10414062/

[9] E. Suryani, L. P. Dewi, L. Junaedi, and R. A. Hendrawan, "A model to improve corn productivity and production," *Journal of Modelling in Management*, vol. 15, no. 2, pp. 589–621, 2019, https://doi.org/10.1108/JM2-11-2018-0181.

[10] L. Castano-Duque, M. Vaughan, J. Lindsay, K. Barnett, and K. Rajasekaran, "Gradient boosting and bayesian network machine learning models predict aflatoxin and fumonisin contamination of maize in Illinois – First USA case study," *Frontiers in Microbiology*, vol. 13, p. 1039947, 2020, https://doi.org/10.3389/fmicb.2022.1039947.

[11] A. M. A. Rahim, A. Sunyoto, and M. R. Arief, "Stroke Prediction Using Machine Learning Method with Extreme Gradient Boosting Algorithm," vol. 21, no. 3, pp. 595–606, 2022, https://doi.org/10.30812/matrik.v21i3.1666.

[12] A. Gafurov, S. Mukharamova, A. Saveliev, and O. Yermolaev, "Advancing Agricultural Crop Recognition: The Application of LSTM Networks and Spatial Generalization in Satellite Data Analysis," *Jurnal Agriculture*, vol. 13, no. 9, p. 1672, 2023, https://doi.org/10.3390/agriculture13091672.

[13] I. P. Putra, R. Rusbandi, and D. Alamsyah, "Klasifikasi Penyakit Daun Jagung Menggunakan Metode Convolutional Neural Network," vol. 2, no. 2, pp. 102–112, 2022, https://doi.org/10.35957/algoritme.v2i2.2360.

[14] D. Iswantoro and D. H. Un, "Klasifikasi Penyakit Tanaman Jagung Menggunakan Metode Convolutional Neural Network (CNN)," *Jurnal Ilmiah Universitas Batanghari Jambi*, vol. 22, no. 2, pp. 900–905, 2022, https://doi.org/10.33087/jiubj.v22i2.2065.

[15] M. Sibiya and M. Sumbwanyambe, "Automatic Fuzzy Logic-Based Maize Common Rust Disease Severity Predictions with Thresholding and Deep Learning," *Journal Pathogens*, vol. 10, no. 2, pp. 1–17, 2021, https://doi.org/10.3390/pathogens10020131.

[16] K. P. Panigrahi, H. Das, A. K. Sahoo, and S. C. Moharana, "Maize Leaf Disease Detection and Classification Using Machine Learning Algorithms," in *Progress in Computing, Analytics and Networking*, H. Das, P. K. Pattnaik, S. S. Rautaray, and K.-C. Li, Eds., vol. 119. Springer, 2020, pp. 659–669.

[17] Q. N. Azizah, "Klasifikasi Penyakit Daun Jagung Menggunakan Metode Convolutional Neural Network AlexNet," *sudo Jurnal Teknik Informatika*, vol. 2, no. 1, pp. 28–33, 2023, https://doi.org/10.56211/sudo.v2i1.227.

[18] G. Kim, Y. Jo, H. Cho, H.-s. Min, and Y. Park. Learning-based screening of hematologic disorders using quantitative phase imaging of individual red blood cells. https://doi.org/10.1101/091983.

[19] D.-y. Wen, J.-m. Chen, Z.-p. Tang, J.-s. Pang, Q. Qin, L. Zhang, Y. He, and H. Yang, "Noninvasive prediction of lymph node metastasis in pancreatic cancer using an ultrasound-based clinicoradiomics machine learning model," *BioMedical Engineering OnLine*, vol. 23, no. 1, p. 56, 2024, https://doi.org/10.1186/s12938-024-01259-3.

[20] Z. Tu, H. Talebi, H. Zhang, F. Yang, P. Milanfar, A. Bovik, and Y. Li, "MaxViT: Multi-axis Vision Transformer," in *Computer Vision – ECCV 2022*, S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, Eds. Springer Nature Switzerland, 2022, vol. 13684, pp. 459–479, https://doi.org/10.1007/978-3-031-20053-3_27.

[21] V. S. Dhaka, S. V. Meena, G. Rani, D. Sinwar, K. Kavita, M. F. Ijaz, and M. Woźniak, "A Survey of Deep Convolutional Neural Networks Applied for Prediction of Plant Leaf Diseases," *Sensors*, vol. 21, no. 14, p. 4749, 2021, https://doi.org/10.3390/s21144749.

[22] Z. Lv and Z. Zhang, "Research on plant leaf recognition method based on multi-feature fusion in different partition blocks," *Journal Digital Signal Processing*, vol. 134, p. 103907, 2023, https://doi.org/10.1016/j.dsp.2023.103907.