Journal of Artificial Intelligence and Engineering Applications

Website: https://ioinformatic.org/

15th October 2025. Vol. 5. No. 1; e-ISSN: 2808-4519

Classification of Pests in Spinach Plants using the Convolutional Neural Network Method

Jefrianus Rohi[1*], Fajar Hariadi[2], Novem Berlian Uly[3]

[1,2,3] *Informatics Engineering Study Program, Wira Wacana Christian University Sumba*
Jefrianusrohi25@gmail.com [1*], fajar@unkriswina.ac.id [2], novemuly@unkriswina.ac.id [3]

Abstract

Agriculture is a key sector that supports food security. Spinach plays an important role in meeting the nutritional needs of communities. However, the productivity of spinach plants in Malumbi Village has decreased due to pest attacks that have not been handled optimally by farmers. Lack of access to information leads to errors in pest identification and handling. Therefore, this study aims to develop a web-based classification system using the Convolutional Neural Network (CNN) method to assist farmers in recognizing pest types in spinach plants quickly and accurately through leaf imagery. CNNs were chosen for their ability to recognize complex visual patterns and generate accurate classifications based on imagery. The research stages include dataset collection, image preprocessing, CNN model training, and accuracy measurement. This research is expected to provide benefits in simplifying the process of digital pest identification, improving the accuracy of diagnosis, and assisting farmers in making more effective pest management decisions. The test results showed that the CNN model built was able to classify pest types with an accuracy rate of up to 84% in the validation data, and was successfully implemented in the form of a web application that could be used directly by farmers.

Keywords: *Convolutional Neural Network,* Spinach, Pests, Image Classification

1. Introduction

Indonesia as an agricultural country has great potential in the production of spinach, one of the vegetables that is widely cultivated because of its short harvest period. However, spinach production is often disrupted by pest infestations such as caterpillars (*Spodoptera litura*, *Plusia*and *Hymenia*) as well as aphids such as *Myzus persicae* and *Thrips* [1]. This pest attack causes damage to the leaves in the form of holes, curling leaves, or spots that reduce the quality and quantity of crops [2].

Malumbi Village, East Sumba Regency, has great potential in spinach cultivation. Almost all residents of Malumbi Village, with a population of around 2,141 people, cultivate spinach vegetables. About half of the total population actively manages land with a total area of about 1,500 hectares spread across several agricultural areas. Each plot has an average area of 1,000 m², and under normal conditions without pest infestation, farmers can produce around 500–1,000 bunches of spinach per harvest. Pest attacks can reduce spinach yields by up to 200–300 bunches per crop, which has a direct impact on farmers' income and economic well-being [3]. In one hectare there are 10 agricultural lands with an average yield of 750 bunches per land, so under normal conditions the total production reaches 7,500 bunches. However, when a pest attack occurs, production can drop by up to 60%, which means that only about 3,000 bunches per hectare remain. This decline certainly has a significant impact on farmers' income and local economic resilience. The main pest that attacks spinach plants in Malumbi Village is armyworms *(Spodoptera litura)* and slugs. The high rate of pest attacks is exacerbated by the lack of access to information for farmers in identifying and handling attacks appropriately [4].

To overcome these problems, solutions are needed that can help farmers identify disturbances quickly and accurately. One of the technologies that can be used is a classification system based on digital images. This technology is part of artificial intelligence designed to recognize visual patterns from images [4]. In this context, the method used is *Convolutional Neural Network* (CNN) [4], which has been shown to be very effective in the recognition of visual patterns in plant imagery for classification purposes. Developing a pest-based classification system *Convolutional Neural Network* (CNN) which is devoted to spinach plants. This system is designed to be able to recognize the symptoms of pest attacks from spinach leaf imagery automatically and accurately. With CNN's ability to extract complex visual features from images, the system built is expected to assist farmers in the process of identifying pests more quickly, accurately, and efficiently [4].

2. Literature Review

2.1. Spinach

Spinach (Amaranthus spp.) is one of the important vegetable crops in Indonesia because of its role as a food source and economic support for farmers. Spinach can grow well in a variety of environmental conditions, both in lowlands and highlands, and is known as an easy-to-cultivate annual. [1].

In addition to being good adaptable to Indonesia's tropical environment, spinach also has a short growth cycle, which is about 25-30 days from planting to harvest. This allows farmers to harvest several times a year, so spinach becomes a commodity that can help stabilize farmers' income in rural areas.[5]

2.2. Pests on spinach leaves

Spinach plants can be attacked by various types of pests that can damage crops and reduce agricultural yields. Some pests that often attack spinach leaves include

1. Armyworm (Spodoptera litura)

   Armyworms are one of the pests that are very detrimental to spinach plants. This pest belongs to the family of 7 Noctuidae and is often found on vegetable plants, including spinach. The armyworm feeds on leaf tissue, leaving large holes in the leaf surface. This damage reduces the efficiency of photosynthesis because the injured leaves are unable to perform photosynthesis properly. Armyworm attacks can accelerate the death of spinach plants if not controlled immediately, as the plant loses many leaves that serve as a place for photosynthesis. Control can be carried out by using insecticides or by natural means, such as introducing natural enemies (for example, parasitoids or natural predators of caterpillars).

2. Snail

   Snails are one of the pests that often attack spinach plants, especially on young leaves that are still soft and fragile. This pest is known as a small tick that sucks fluid from leaf tissue using a needle-shaped mouth (stylet). Sifut infestation causes spinach leaves to become wrinkled, yellowish, wilted, and can ultimately inhibit the overall growth of the plant. Not only that, sifut also acts as a vector for spreading viruses, such as mosaic viruses, which aggravates damage to spinach plants. Sifut pest control can be done by spraying insecticides or by natural approaches, such as using natural enemies in the form of spiders or predatory insects such as ladybird beetles (Coccinellidae).

2.3. Digital Image

Digital imagery is a visual representation of an object or scene that has been converted into numerical form so that it can be stored, analyzed, and manipulated using a computer [6]. In image processing, visual information is broken down into the smallest units called pixels, which are small square-shaped image elements that each store a numerical value. This value reflects the intensity of brightness or color at a given location in a two-dimensional plane.

2.4. Metode Convolutional Neural Network

Convolutional Neural Network (CNN) is one of the most widely used deep learning architectures in digital image processing. CNN is specifically designed to automatically recognize visual patterns from image data through a layered learning process. In the context of image processing, CNNs have the ability to extract important features from images, such as shapes, colors, textures, and other spatial patterns, which are useful in the process of classification and identification of objects [7].

CNN as a sophisticated machine learning model has become the standard for a wide range of applications, including visual identification, medical image analysis, image segmentation, and natural language processing (NLP). CNNs are specifically designed to process two-dimensional data, such as images, so that they can extract relevant local features more efficiently than other methods. CNN is therefore more effective because it can automatically extract important features from the input data without the need for human intervention [8]. CNN consists of several tiers that include the Convolutional Layer, the Pooling Layer, and the Fully Connected Layer, as shown in the following figure.

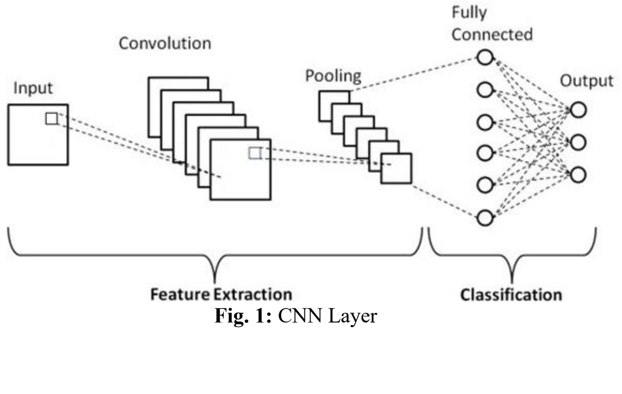


Fig. 1: CNN Layer

2.5. Web

A website is a collection of digital pages that contain information in the form of text, images, animations, sounds, and videos or a combination of all of them that are presented over an internet connection and can be accessed globally using HTML, and are accessible to anyone connected [9].

In web-based application development, two main categories are known, namely static web and dynamic web. A static web is a page whose content is fixed and can only be changed directly by the developer. In contrast, dynamic web has the ability to display content that changes automatically based on user interaction or data from the server. In this study, a CNN-based pest classification application was developed using a dynamic web approach, as it allows users (farmers) to upload images of spinach leaves and receive direct diagnosis results based on the images [10].

2.6. Waterfall method

The method used in the development of this system is the Waterfall method, which consists of sequential stages ranging from needs analysis to system evaluation. The following are the stages in the Waterfall model applied in this study:

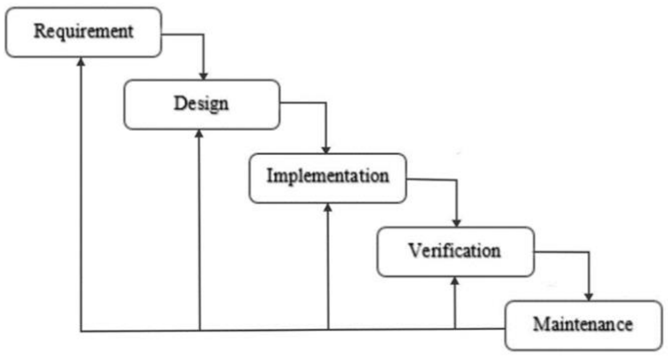


Fig. Error! No text of specified style in document.**2:** Waterfall Method

3. Methodology

3.1. Profile of the research object

Malumbi Village is one of the areas located in Kambera District, East Sumba Regency, East Nusa Tenggara Province. This village has a population of around 2,141 people. Most of the Malumbi people make a living as farmers, with a main focus on vegetable cultivation, especially spinach (Amaranthus spp.), which is a local leading commodity. The research flow carried out in making CNN-based classification for the diagnosis of pests on spinach leaves in Malumbi Village is as follows:

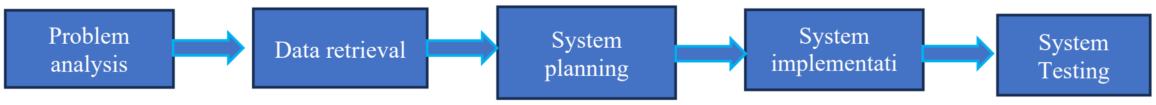


Fig. 3: Research flow

Figure 3 shows the research flow consisting of the first problem analysis: 1. The research begins by identifying the main problem in the field, namely the difficulty of farmers in identifying pest types quickly and accurately. This condition causes mishandling and crop losses. 2. Data collection stage: at this stage, data collection is carried out by means of observation, interviews and documentation to obtain the required information. 3. CNN stage analysis: At this stage, pre-image processing and CNN model creation is carried out. The steps cover RGB to grayscale conversion, Image resize to standard size, Feature extraction with convolution layer, ReLU Activation, Pooling for dimension reduction, Flatten and classification with fully connected layers. 4. System (web) development: This classification system is wrapped in the form of PHP (native) and Bootstrap-based web applications. The application allows users (farmers) to upload leaf images to get direct diagnosis results. 5. Implementation: The application is tested in real conditions to see the performance of the classification and its usefulness in helping farmers in the field. 6. System testing: Testing is carried out to ensure that all system functions run according to needs, both technically (black-box testing) and from the user side (ease of use and relevance of results).

3.2. CNN model creation

The creation of the Convolutional Neural Network (CNN) model began with image preprocessing and continued with CNN model training for the classification of spinach leaf images. The following are the stages carried out:

3.2.1. Image Preprocessing

Before the image is fed into the CNN model, the image must be processed first. Preprocessing aims to prepare data so that the CNN model can process it efficiently.

3.2.2. Convert RGB images to Grayscale

The original images collected are in RGB format, consisting of three color channels: Red (R), Green (G), and Blue (B). However, to simplify the calculations and reduce complexity, the image is converted to a grayscale image. This conversion process is done using a standard formula that combines the contributions of all three color channels, the Conversion to Grayscale Formula.
Grayscale = (0.2989 * R) + (0.5870 * G) + (0.1140 * B)
This formula is used to convert each pixel from an RGB image to grayscale, with a specific weight for each color channel (Red, Green, Blue). Here is an example of a table of intensity values from a spinach leaf image (5×5 pixels):

Table 1: RGB Matrix

Red					Green					Blue				
43	40	41	138	128	103	100	101	164	154	31	28	29	119	109
44	43	46	138	129	104	103	106	164	155	32	31	34	119	110
43	43	49	143	135	105	105	111	166	158	32	32	38	122	114
135	134	134	104	104	170	169	169	159	159	76	75	75	40	40
133	134	134	104	104	168	169	169	159	159	74	75	75	40	40

The table above shows the color intensity values of the spinach leaf imagery in RGB format. Each row represents one row of pixels in the image, while the columns display the intensity value for each color channel.
The following table presents the intensity values of each color channel (R, G, and B) in the spinach leaf image in a 5×5 matrix, where each cell shows the value of the color channel for one pixel at a specific position in the image:

Table 2: Matrix Data (5x5)

Baris	R	G	B
1	43, 40, 41, 138, 128	103,100,101,164,154	31, 28, 29, 119, 109
2	44, 43, 46, 138, 129	104,103,106,164,155	32, 31, 34, 119, 110
3	43, 43, 49, 143, 135	105,105,111,166,158	32, 32, 38, 122, 114
4	135,134,134,104,104	170,169,169,159,159	76, 75, 75, 40, 40
5	133,134,134,104,104	168,169,169,159,159	74, 75, 75, 40, 40

Table 2 shows the color intensity values of each channel in the 5×5 pixel spinach leaf image. The color channels displayed include *Red* (R), *Green* (G), and *Blue* (B). Each row in the table represents the row of pixels in the image, while each column presents the intensity value of each channel at the same pixel position. These values range from 0 to 255, which is a standard representation in an 8-bit digital system. These values will be used in the grayscale conversion process to produce one single intensity value per pixel.

3.2.3. Resize Image

Once the image is converted to grayscale, it will be resized to ensure a consistent size, usually to a smaller size such as 64x64 or 128x128 pixels, to be more efficiently processed by CNN. This resize process involves recalculating the pixel values to fit the new size. For example, if the size of the original image is 5x5 and we want to resize it to 4x4, then we'll calculate the average pixel value of each block to produce one value on a smaller image. This process also aims to reduce complexity and computational time, while keeping important information out of the image. Divide the 5×5 matrix into 9 blocks (each), then calculate the average of each block

$$I'_{(i,j)} = \frac{1}{k\ x\ k}\sum_{x=0}^{k-1}\sum_{y=0}^{k-1} I(i+x, j+y)$$

Information:

I	: the pixel intensity value matrix of the initial image (grayscale)
I′	: matrix results resize (smaller size).
i,j	: the position of the block (rows and columns) in the resize result.
K×K	: Block size 2×2).
x,y	: the index in blocks (0 or 1 if the block is 2×2).

3.3. Stages of the CNN model

3.3.1. Convolutional Layer

Convolution layers aim to extract features from images through 2D convolution operations using filters (kernels). This process is done by multiplying the elements in an image block (e.g. 2×2) by the corresponding kernel elements, and then summing the result.
The mathematical formula is written as:

$$O_{(i,j)} = \frac{1}{k\ x\ k}\sum_{m=0}^{k-1}\sum_{n=0}^{k-1} I(i+m, j+n).(m,n)$$

Where	:
I	: citra input
K	: kernel 2D,
O (I, j)	: Output (feature map) in position (i, j)
m,n	: The index of elements in the kernel.
K	: The kernel side size (kernel 2×2).

3.3.2. Activation Layer

After a convolutional operation, an activation function is usually applied to each element of the generated feature map. The activation function introduces non-linearity into the model, which allows CNN to study more complex patterns. One of the commonly used activation functions is ReLU (*Rectified Linear Unit*). ReLU Activation Function Formula: f(x) = max (0, x). This function converts all negative values to 0 and maintains positive values. Input to the ReLU (convolutional result feature map).

$$\text{Matrix result } \textit{of featur map} = \begin{bmatrix} 150 & 152 & 229 \\ 155 & 182 & 255 \\ 227 & 251 & 276 \end{bmatrix}$$

$$\text{Outcome Matrix after ReLU} = \begin{bmatrix} 150 & 152 & 229 \\ 155 & 182 & 255 \\ 227 & 251 & 276 \end{bmatrix}$$

All values in the *feature map matrix* are already positive, the application of the ReLU function does not change these values. Therefore, the outcome matrix after the ReLU activation function remains the same.

3.3.2. Pooling Layer

The pooling layer aims to reduce the spatial dimension of the feature map, thereby reducing the number of parameters and computations in the network, as well as making the features more invariant to small changes in input positions. The two common types of pooling are *Max Pooling* and *Average Pooling.*

Max *Pooling formula* (pool size 2x2, stride 1) for each 2x2 area on the input, the output is the maximum value within that area.

$$y = max\{\text{xi, j} \mid \text{i} = 1,2; \text{j} = 1,2\}$$

Where x i, j are the elements in the 2×2 area.

3.3.3. Flatten *Layer*

After several layers of convolution and pooling, *the multidimensional feature map* needs to be converted into a one-dimensional vector before it enters the *fully connected* layer. The *flatten layer* takes all the elements of the feature map and arranges them into a single column vector. Example Manual calculation using the output from the previous pooling layer (for example, the result of Max Pooling 1x1).

$$\text{Matrix } Output_{Pooling} = \begin{bmatrix} 182 & 255 \\ 251 & 276 \end{bmatrix}$$

The *flatten* layer will convert the matrix into a one-dimensional vector as follows:

Vector = [182, 255, 251, 276]

Thus, the data from the *feature map* in the form of a two-dimensional matrix is successfully converted into a vector form that can be processed by *the fully connected* layer for further classification or prediction

3.3.5. Fully *Connected Layer*

A fully connected layer (or dense layer) is a layer where each neuron is connected to every *neuron* in the previous layer. This layer is used to classify based on features that have been extracted by previous layers.

General formula of neuron output:

$$z_i = \sum_{k=1}^{n} x_k \, . w_{k,i} + b$$

Where :

xk	: input to neuron (input vector element)
wk,i	: the weight of the input to the i-i neuron
Bi	: Neuron bias of I-I
Zi	: linear value of the i neuron
f(z)	: activation function (e.g. ReLU or softmax)

Then the output value of the neuron is calculated by the activation function:

hi = f(zi)

3.4. Web Application

The creation of a web application is one of the important stages in this study which functions as a medium of interaction between users and the pest classification system on spinach leaves based on the Convolutional Neural Network (CNN) method. This application is designed to be easily used through computer devices or mobile devices connected to the internet network. This web application is built using the native PHP programming language on the backend, as well as HTML5, CS3, JavaScript, and Bootstrap on the frontend. The choice of this technology was made because it takes into account ease of implementation, high compatibility with various hosting platforms, and efficiency in the development of a responsive user interface.

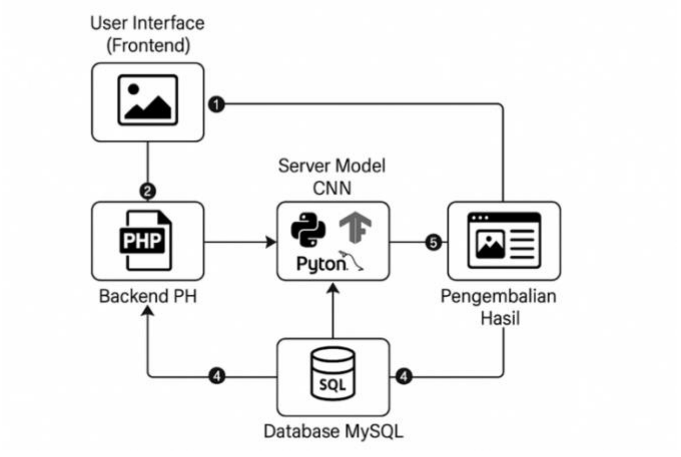


Fig. 4: Web Application System Architecture (Frontend Backend)

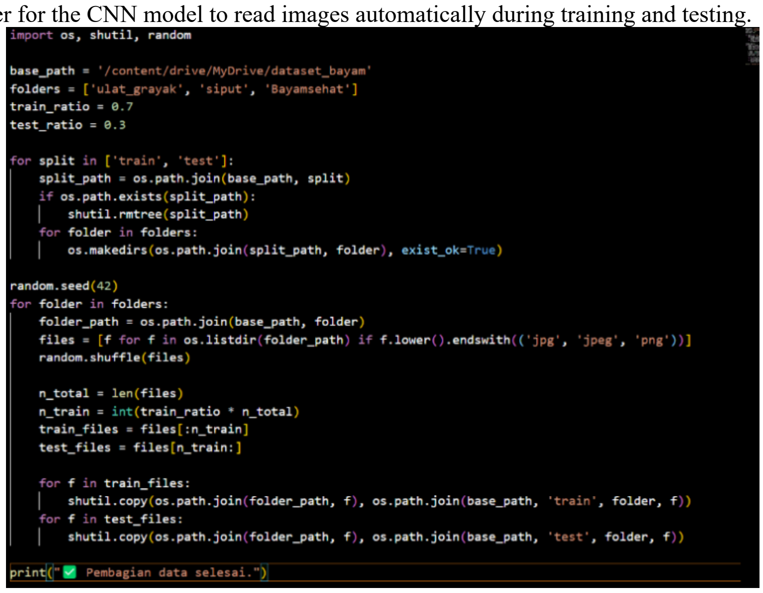
4. Results and conclusions

4.1. Analysis and implementation of the CNN model

The CNN model was implemented using Google Colaboratory with Python language as well as the TensorFlow and Keras libraries. The process includes data capture, image preprocessing, architecture design, training, evaluation, and integration into web systems. The dataset consists of 300 images of spinach leaves divided into three classes: healthy spinach, armyworm infested, and snail-infested. The data was collected directly from agricultural land in Malumbi Village, East Sumba. The dataset is divided into 70% training data and 30% test data randomly and randomly for each class. The model is designed to recognize distinctive visual patterns on leaves, both healthy and pest-damaged, with the aim of generating accurate and useful classifications in support of digital agriculture.

4.1.1. Dataset Sharing

The collected spinach leaf imagery dataset was organized into three main folders based on class labels: healthy spinach, spinach attacked by armyworms, and spinach attacked by snails. For model training purposes, the dataset is automatically divided into two parts: 70% for training data and 30% for testing data. This process is done randomly and evenly by using the help of programming in Google Colaboratory.
The folder structure is tailored to the format required by the TensorFlow library, where each class folder is placed in a train and test directory. This makes it easier for the CNN model to read images automatically during training and testing.

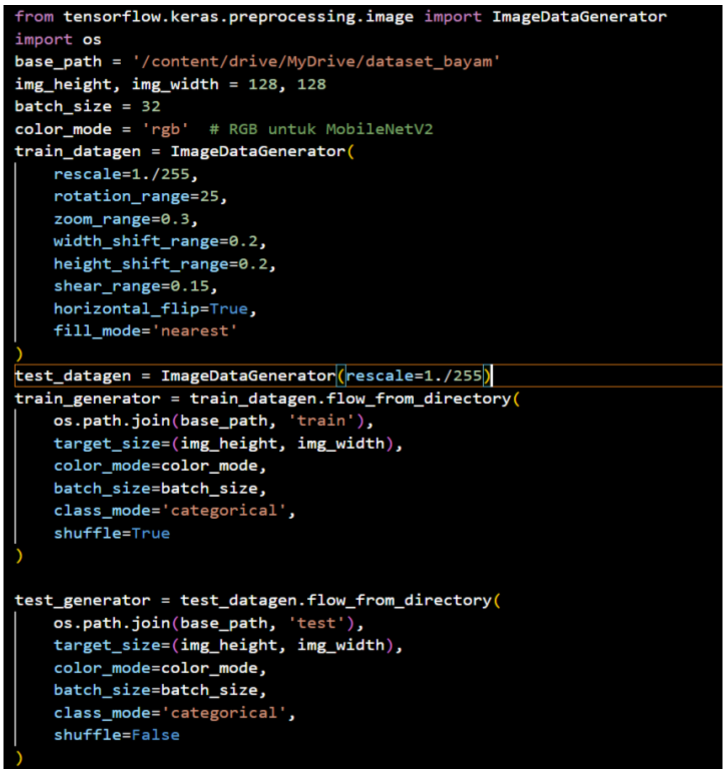


```
import os, shutil, random

base_path = '/content/drive/MyDrive/dataset_bayam'
folders = ['ulat_grayak', 'siput', 'Bayamsehat']
train_ratio = 0.7
test_ratio = 0.3

for split in ['train', 'test']:
    split_path = os.path.join(base_path, split)
    if os.path.exists(split_path):
        shutil.rmtree(split_path)
    for folder in folders:
        os.makedirs(os.path.join(split_path, folder), exist_ok=True)

random.seed(42)
for folder in folders:
    folder_path = os.path.join(base_path, folder)
    files = [f for f in os.listdir(folder_path) if f.lower().endswith(('jpg', 'jpeg', 'png'))]
    random.shuffle(files)

    n_total = len(files)
    n_train = int(train_ratio * n_total)
    train_files = files[:n_train]
    test_files = files[n_train:]

    for f in train_files:
        shutil.copy(os.path.join(folder_path, f), os.path.join(base_path, 'train', folder, f))
    for f in test_files:
        shutil.copy(os.path.join(folder_path, f), os.path.join(base_path, 'test', folder, f))

print("✅ Pembagian data selesai.")
```

Fig. 5: Spinach Leaf Image Dataset Sharing Code in Google Colaboratory

4.1.2. Image Preprocessing and Augmentation

The imagery used in the CNN model training undergoes several stages of preprocessing in order to be optimally recognized by the network architecture. The entire image is resized to a resolution of 128x128 pixels, and the color scheme is kept in RGB format to be compatible with the MobileNetV2 architecture used. In order for the model to be able to learn from various variations in the shape and position of the leaf image, an augmentation process is carried out which includes: Image rotation, Zooming, Width and height shift, Shearing, Horizontal mirroring.



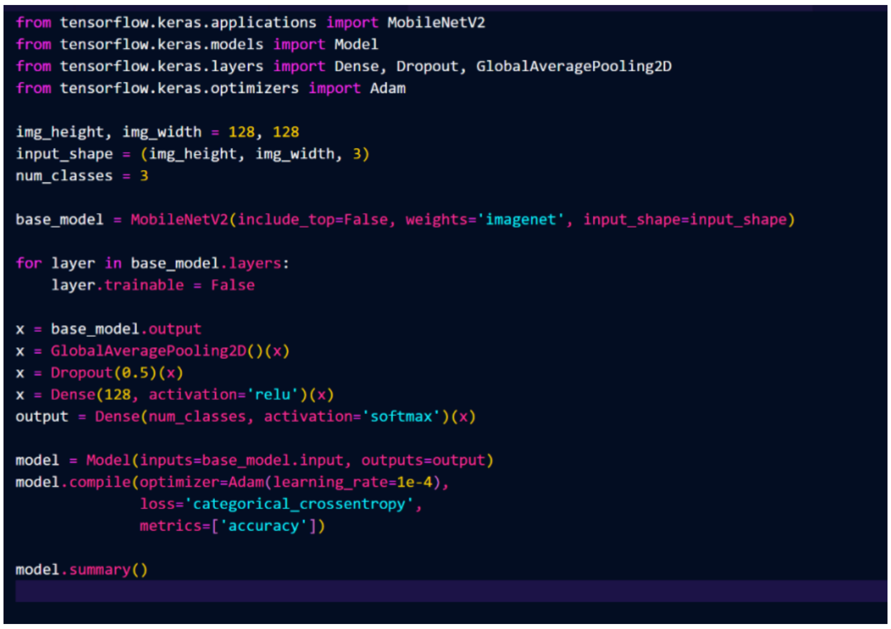
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import os
base_path = '/content/drive/MyDrive/dataset_bayam'
img_height, img_width = 128, 128
batch_size = 32
color_mode = 'rgb'  # RGB untuk MobileNetV2
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=25,
    zoom_range=0.3,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.15,
    horizontal_flip=True,
    fill_mode='nearest'
)
test_datagen = ImageDataGenerator(rescale=1./255)
train_generator = train_datagen.flow_from_directory(
    os.path.join(base_path, 'train'),
    target_size=(img_height, img_width),
    color_mode=color_mode,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=True
)

test_generator = test_datagen.flow_from_directory(
    os.path.join(base_path, 'test'),
    target_size=(img_height, img_width),
    color_mode=color_mode,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)
```

Fig. 6: Code Augmentation and Preprocessing Data Training and Testing with ImageDataGenerator

4.1.3. CNN Model Architecture Design Using MobileNetV2

In this study, a transfer learning approach was used using the MobileNetV2 architecture as a basic model. MobileNetV2 was chosen because it has high efficiency in terms of size and speed, yet still provides competitive classification performance, especially for devices with limited resources. The MobileNetV2 model used is a pretrained version of the ImageNet dataset, and is modified by disabling (freezing) the entire initial convolutional layer. This aims to retain the general features that have been learned from the large dataset, so that only the top of the model will be trained according to the spinach leaf dataset.
On top of the basic model, several layers are added, namely:

1. GlobalAveragePooling2D, to flatten the spatial features of the image,
2. Dropout by 50%, to reduce the risk of overfitting,
3. Dense layer with 128 units and ReLU activation, to extract advanced features,
4. Dense output layer with 3 units (according to the number of classes) and softmax activation, to perform multiclass classification.



```python
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Dropout, GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam

img_height, img_width = 128, 128
input_shape = (img_height, img_width, 3)
num_classes = 3

base_model = MobileNetV2(include_top=False, weights='imagenet', input_shape=input_shape)

for layer in base_model.layers:
    layer.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dropout(0.5)(x)
x = Dense(128, activation='relu')(x)
output = Dense(num_classes, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=output)
model.compile(optimizer=Adam(learning_rate=1e-4),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.summary()
```

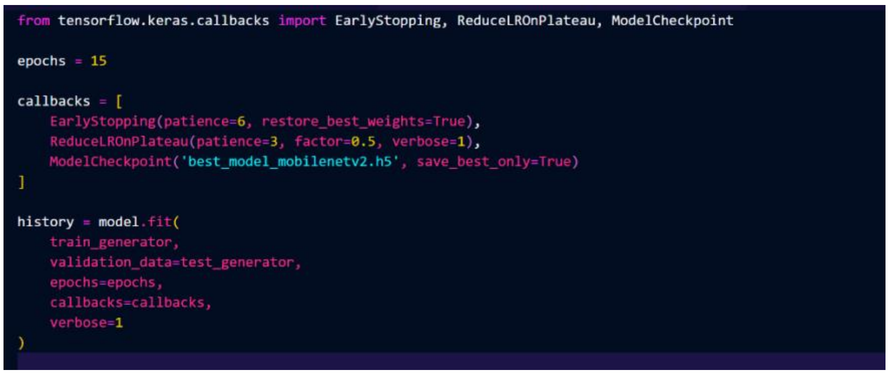
Fig. 7: CNN Architecture

4.1.4. CNN Model Training

The CNN model training process was carried out at the Google Colaboratory for 15 epoches using a training and test data generator. During the training, several important callbacks were implemented to optimize the process, namely:

1. EarlyStopping with patience 6 to stop training early if there is no performance improvement,
2. ReduceLROnPlateau to lower the learning rate if the model does not show improvement in some epochs,
3. Checkpoint model to store the best model based on accuracy validation during the training process.

The model shows a trend of gradual improvement in accuracy from the beginning to the end of the training. In the first epoch, the validation accuracy was recorded at 58.89%, and continued to increase until it reached 84.44% in the 15th epoch. At the same time, the validation loss value continued to decrease from 0.9213 to 0.3453, indicating that the model learned effectively and stably without overfitting.
The performance improvements were evident with each subsequent epoch. For example, in the 5th epoch the validation accuracy has reached 80%, then it is consistently above 81% until the end of the training. The highest accuracy was achieved in the 12th to 15th epoch, where the validation accuracy was stable at 84.44%.
The evaluation was carried out using test data as many as 90 images, and the results showed that the model had a good level of generalization to data that had never been seen before. The evaluation accuracy value obtained is consistent with the validation accuracy during the training.



```python
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint

epochs = 15

callbacks = [
    EarlyStopping(patience=6, restore_best_weights=True),
    ReduceLROnPlateau(patience=3, factor=0.5, verbose=1),
    ModelCheckpoint('best_model_mobilenetv2.h5', save_best_only=True)
]

history = model.fit(
    train_generator,
    validation_data=test_generator,
    epochs=epochs,
    callbacks=callbacks,
    verbose=1
)
```

Fig. 8: CNN Model Training

4.1.5. Accuracy and Loss Charts

During the CNN model training process, accuracy and loss were recorded in training data and testing data in each epoch. The results of the training are visualized in the graph

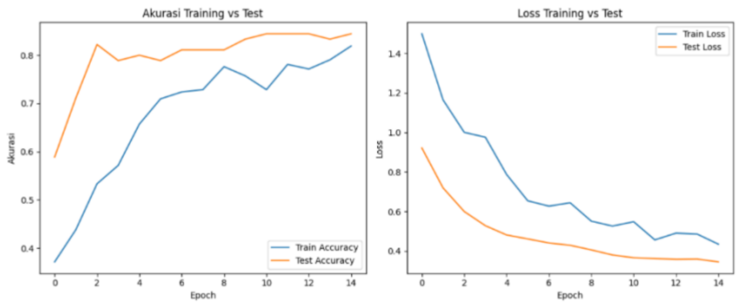


Fig. 9: Training vs Testing Accuracy Graph and Training vs Testing Loss

In Figure, it can be seen that the accuracy value in the training data has increased consistently as the epoch increases, although there is a slight fluctuation. Meanwhile, the accuracy of the test data was stable above 80%, indicating that the model was able to generalize well to data that had never been seen. It shows that the loss value on the training and test data decreases significantly during the training process. The balanced reduction in loss between the training data and the test data indicates that the model is not overfitting, and indicates an optimal training process.

4.1.6. Evaluation of the CNN Model

After the training process was completed, the CNN model was evaluated using 90 test data, 30 images each for each class: healthy spinach, snail-infested, and armyworm-infested. The results of the evaluation showed that the model had an accuracy of 84.44% with a loss value of 0.3453 on the test data. Further evaluation uses a confusion matrix and classification report to measure the model's performance in detecting each class.

Table 3: Confusion Matrix Model CNN

Current \ Prediction	Healthy	Siput	Caterpillar
Healthy	30	0	0
Siput	0	25	5
Grayworm	0	9	21

Table 4: Precision, Recall, dan F1-Score Model CNN

Class	Precision	Recall	F1-Score	Support
Healthy spinach	1.00	1.00	1.00	30
Siput	0.74	0.83	0.78	30
Grayworm	0.81	0.70	0.75	30
Average	0.85	0.84	0.84	90

From the confusion matrix and classification report above, it can be concluded that the model has good classification ability for healthy spinach leaves, with perfect precision and recall. Although the performance against the snail and armworm classes was slightly lower, the model was still able to detect both types of pests quite well, as seen from the f1-score above 0.70 for both.

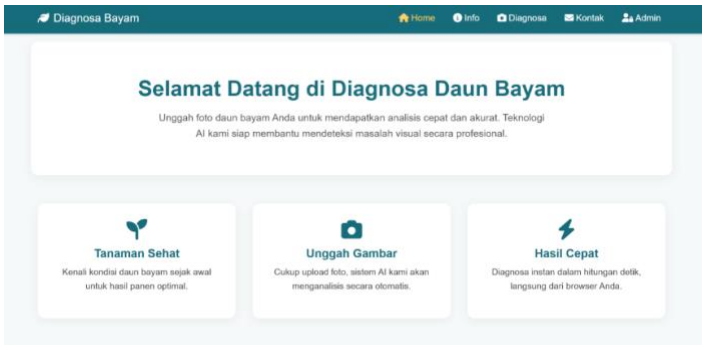
4.2. System Planning

The design of the system is carried out to describe the structure, processes, and interfaces of the system that are built before the implementation stage. This system is designed to be able to automatically classify pests in spinach plants through leaf imagery using the Convolutional Neural Network (CNN) method. The system has two types of users, namely:

1. User: can access information features, spinach leaf diagnostics, and contacts.
2. Admin: has additional access such as managing diagnostic data, user data, and can add new user accounts

4.3. Implementation

The implementation of the system is carried out by integrating CNN models that have been trained using Google Colab into a PHP-based web application. The model_bayam_keras.h5 model file is downloaded and called using a 64 Python script connected to PHP. The web is built using XAMPP, HTML, and Bootstrap, with the main features of spinach leaf image upload pages, classification processing, and diagnostic results display. PHP calls Python via shell_exec() to run the classification and display the results to the user.



Selamat Datang di Diagnosa Daun Bayam

Unggah foto daun bayam Anda untuk mendapatkan analisis cepat dan akurat. Teknologi AI kami siap membantu mendeteksi masalah visual secara profesional.

Kenali kondisi daun bayam sejak awal untuk hasil panen optimal.

Cukup upload foto, sistem AI kami akan menganalisis secara otomatis.

Diagnosa instan dalam hitungan detik, langsung dari browser Anda.

Fig. Error! No text of specified style in document.**10:** Home Page

This page is the initial page that is displayed when the user opens the system. Provides a brief description of the system, benefits, and navigation menus to other features.

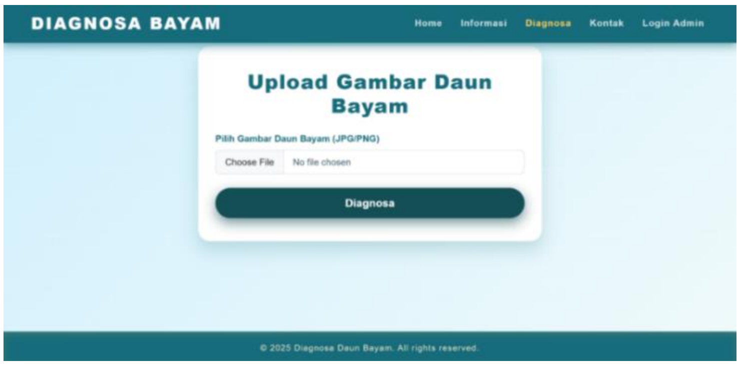
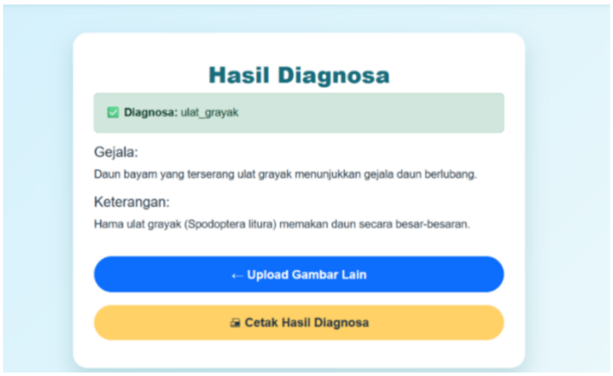


Fig. 11: Menu Diagnosis

On this page, users can upload images of spinach leaves. Once the image is uploaded, the system will display the prediction results based on the images analyzed by the CNN model.



✓ **Diagnosa:** ulat_grayak

Gejala:

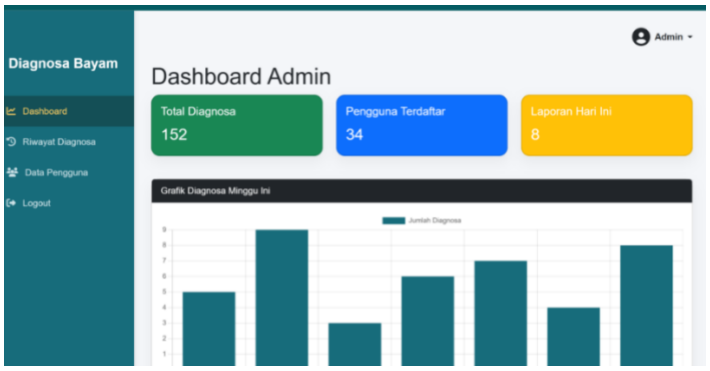
Daun bayam yang terserang ulat grayak menunjukkan gejala daun berlubang.

Keterangan:

Hama ulat grayak (Spodoptera litura) memakan daun secara besar-besaran.

Fig. 12: Diagnosis Results

On this page is the diagnostic result, after the user uploads a picture of the spinach leaf that is infested with pests, the system will display the diagnostic results based on the image above.



Dashboard Admin

Total Diagnosa 152

Pengguna Terdaftar 34

Laporan Hari Ini 8

Fig. 13: Dashboard Admin

The dashboard displays diagnostic statistics, number of users, and classification history. Admins can also manage user data and add new accounts.

4.4 System Testing

The system test was carried out using the black-box method, by testing the functions of uploading images, automatic classification, and output results to the screen. All features are tested with valid and invalid inputs. The results show that all components are running well**.**

1. Conclusion

Based on the results of research that has been carried out regarding the classification of pests in spinach plants using the Convolutional Neural Network (CNN) method, the following can be concluded

1. Development of a Pest Classification Application System This research has succeeded in developing a web-based pest classification application on spinach plants using the *Convolutional Neural Network* (CNN) method. The system built allows users, especially farmers in Malumbi Village, to upload images of spinach leaves and get the results of identifying pest types automatically. The application was built using PHP for the web interface, and the CNN model was trained using a spinach leaf imagery dataset on the Google Colab platform, then integrated into the web system in the form of a file.

2. CNN Model Performance in Image Classification The CNN model developed is able to classify spinach leaf imagery into three classes, namely: healthy spinach, snail-infested, and armyworm-infested. Based on the test results, the CNN model showed a validation accuracy rate of 84.44%, with the highest precision and recall values in the "bayam_sehat" class. These results show that the model has a fairly good performance in recognizing visual symptoms of pests in spinach leaves. However, there is a slight imbalance in the performance of the "ulat_grayak" class classification, which can be improved in the future by adding a variety of training data.

References

[1] J. B. Xavier, D. B. de Andrade, D. C. de Souza, G. C. Guimarães, L. V. Resende, and R. M. Guimarães, “Morphological, chemical and physiological characterization of amaranthus spp. Seeds,” *J. Seed Sci.*, vol. 41, no. 4, pp. 478–487, 2019, doi: 10.1590/2317-1545v41n4226286.

[2] Patel and R. Goyena, "Spinach as a Farmer's Sustain Vegetable in Indonesia," *J. Chem. Inf. Model.*, vol. 15, no. 2, pp. 9–25, 2019.

[3] A. M. Lesmana, R. P. Fadhillah, and C. Rozikin, "Disease Identification in Potato Leaf Images Using Convolutional Neural Network (CNN)," *J. Science and Inform.*, vol. 8, no. 1, pp. 21–30, 2022, doi:10.34128/jsi.v8i1.377.

[4] A. Kamilaris and F. X. Prenafeta-Boldú, “Deep learning in agriculture: A survey,” *J. Comput. Electron. Agric.*, vol. 147, no. July 2017, pp. 70–90, 2018, doi: 10.1016/j.compag.2018.02.016.

[5] L. A. Rambe, W. Lestari, Y. Triyanto, and Y. Sepriani, “Optimizing Red Spinach (Amaranthus tricolor) Growth and Yield by Applying Organic and Inorganic Fertilizers,” *J. Agron. Tanam. Too much.*, vol. 6, no. 2, 2024, doi: 10.36378/juatika.v6i2.3624.

[6] S. V Khedaskar, M. A. Rokade, B. R. Patil, and T. P. N, “A Survey of Image Processing and Identification Techniques,” *Int. J. Res. Innov.*, vol. 1, no. 1, pp. 1–10, 2018.

[7] L. Zewen, L. Fan, Y. Wenjie, P. Shouheng, and Z. Jun, “A survey of convolutional neural networks: analysis, applications, and prospects,” *IEEE Trans. neural networks Learn. Syst.*, vol. 33, no. 12, pp. 6999–7019, 2021.

[8] M. M. Taye, “Understanding of Machine Learning with Deep Learning :,” *Comput. MDPI*, vol. 12, no. 91, pp. 1–26, 2023.

[9] A. Permatasari and S. Suhendi, "Design and Build a Film Talent Management Information System based on Web Applications," *J. Inform. Terpadu*, vol. 6, no. 1, pp. 29–37, 2020, doi: 10.54914/jit.v6i1.255.

[10] D. W. L. Pamungkas and S. Rochimah, "Web Application Testing - A Systematic Literature Review," *J. Science and Technology*, vol. 23, no. 1, pp. 17–24, 2019, doi: 10.31284/j.iptek.2019.v23i1.459.