Classification of Macronutrient Deficiencies in Melon Leaves Using Convolutional Neural Networks

Raihan Ade Purnomo[1]

[1]Informatics Engineering, Faculty of Science and Technology
[1] Syarif Hidayatullah State Islamic University Jakarta
Jl. Ir H. Juanda No.95, Tangerang Selatan
e-mail: [1]raihan.adepurnomo22@mhs.uinjkt.ac.id

Abstract

Macronutrient deficiencies such as calcium, nitrogen, and potassium shortages in melon plants can significantly reduce harvest quality and productivity. This study aims to develop a classification model for detecting macronutrient deficiencies in melon leaves using Convolutional Neural Network (CNN). The dataset consists of 200 melon leaf images categorized into calcium deficiency, nitrogen deficiency, potassium deficiency, and healthy leaves. Data augmentation techniques were applied to enhance dataset variation, and the model was trained using Google Colab with GPU support. The CNN architecture includes convolutional layers for feature extraction, pooling layers for dimensionality reduction, dropout layers to prevent overfitting, and fully connected layers for classification. The model achieved high accuracy in classifying melon leaf conditions, demonstrating its potential as an effective tool for early detection of nutrient deficiencies. This research contributes to agricultural technology by providing an automated, accurate, and efficient method for diagnosing plant nutritional problems, enabling farmers to take timely preventive actions to improve crop yield and quality.

Keywords: *Convolutional Neural Network, Macronutrient Deficiency, Melon Leaves, Machine Learning.*

1. Introduction

Cucumis melo, commonly known as melon, is one of the horticultural commodities with high economic value. However, the success of melon cultivation is highly influenced by plant nutritional conditions, particularly macronutrients such as calcium, nitrogen, and potassium. Deficiencies in these macronutrients can cause various physiological problems, including leaf yellowing, stunted growth, and reduced quality and quantity of harvest yields [1]. If not detected promptly, nutrient deficiencies can have detrimental effects, resulting in significant economic losses for farmers.

Currently, nutrient deficiency detection is still largely performed manually based on visual observation. This method tends to be subjective and requires specialized expertise, thus not always providing accurate results. Artificial intelligence-based technology offers a potential solution to address this problem [2]. By utilizing plant leaf images, models based on Convolutional Neural Network (CNN) can be used to automatically and accurately detect and classify types of nutrient deficiencies [3].

This study aims to develop a classification model for macronutrient deficiencies in melon leaves using CNN. The model will be trained with a dataset of melon leaf images covering categories of calcium, nitrogen, and potassium deficiencies, as well as healthy leaves as control. By using the Google Colab platform for the training process, this research is expected to produce an efficient and accurate model that can assist farmers in detecting nutrient deficiencies early. Early detection enables farmers to immediately take preventive actions, prevent further damage, and improve the productivity and quality of melon harvests.

2. Research Method / Proposed Method

Figure 1. CRISP-DM Methodology

The methodology used in this research is CRISP-DM (Cross-Industry Standard Process for Data Mining). CRISP-DM is a systematic framework that supports structured data-based project development. The following are the CRISP-DM stages adapted for this research:

2.1. Business Understanding

Starting with understanding the main objectives of the research, which is to develop a classification model capable of automatically detecting macronutrient deficiencies in melon leaves based on leaf images. This research aims to help farmers diagnose nutrient deficiencies such as calcium, nitrogen, and potassium early, thereby improving harvest yields [1]. Macronutrient deficiencies have been known to significantly impact the quality and quantity of melon harvests, which are often difficult to detect manually. By utilizing artificial intelligence technology, especially CNN, this system is expected to provide high accuracy and efficiency in identifying plant symptoms.

2.2. Data Understanding

The Data Understanding stage involves exploring the dataset used, which is 200 melon leaf images taken from Kaggle. This dataset consists of four categories with balanced representation, namely 50 images for each class: calcium deficiency, nitrogen deficiency, potassium deficiency, and healthy category. The data exploration process includes data distribution analysis, pattern visualization, and data quality checks to ensure the dataset is adequate for model training. Gonzalez and Woods [4] state that "understanding the dataset is a crucial step in ensuring the reliability and accuracy of any image classification model."

2.3. Data Preparation

Focused on data cleaning and augmentation to increase dataset variation. This stage includes image normalization and augmentation such as rotation, flipping, and scale changes. Gonzalez et al. [9] emphasize that "feature extraction techniques in digital image processing isolate and quantify visual characteristics, which are critical for classification tasks." This process aims to ensure that the model is able to handle data variations effectively, including changes in position, rotation, or lighting in leaf images.

2.4. Modeling

The Modeling stage includes CNN model development with an architecture adapted for melon leaf image classification. CNN was chosen because of its ability to capture spatial features from images, such as leaf color and texture patterns [5]. The model was trained using the Google Colab platform with GPU to accelerate the training process. Hyperparameter settings, such as the number of filters in convolutional layers and kernel size, were tested to optimize accuracy.

2.5. Evaluation

The Evaluation stage aims to evaluate model performance with metrics such as accuracy, precision, recall, and F1-score. The model is tested using validation data to measure generalization ability to new data. The results of this evaluation determine whether the model meets research needs and is ready for use in the next stage. Springenberg et al. [6] mention that "thorough evaluation of a model ensures that it performs reliably under real-world conditions."

2.6. Deployment

The Deployment stage involves implementing the model into a web-based system to facilitate access by end users. The trained model is integrated into an intuitive application interface, allowing farmers to upload leaf images directly and get diagnosis results in real-time. Szeliski [7] states that "computer vision applications have the potential to transform agricultural practices by providing real-time insights to farmers."

3. Literature Study

3.1 Macronutrients in Plants

Macronutrients are essential elements that support various physiological and metabolic functions in plants, including melon. Nitrogen (N), potassium (K), and calcium (Ca) play vital roles in plant growth and productivity, but deficiencies in these elements can have significant negative impacts [1].

3.1.1 Functions of Macronutrients

Nitrogen (N): Nitrogen is required for chlorophyll, protein, and nucleic acid synthesis. This element promotes vegetative plant growth such as stems, leaves, and roots. Nitrogen also supports photosynthesis which increases overall plant productivity [1]. Balanced nitrogen application helps plant organ growth and increases photosynthesis yield.

Potassium (K): Potassium functions as a regulator in plant metabolism, including photosynthesis, protein synthesis, and carbohydrate formation. This element also strengthens plant tissues, making them more resistant to drought, pests, and diseases [1]. Potassium improves fruit quality by preventing flower and fruit drop [8].

Calcium (Ca): Calcium helps form cell walls and improves cell membrane permeability. Additionally, this element supports root hair growth and increases the efficiency of absorption of other nutrients [1]. Calcium also helps neutralize toxic compounds in the soil [8].

3.1.2 Impact of Macronutrients Deficiency

Macronutrient deficiency has significant impacts on the physiology and productivity of melon plants. Nitrogen deficiency causes leaf yellowing due to reduced chlorophyll synthesis, stunted plant growth, and decreased photosynthetic capacity, resulting in suboptimal harvest yields [1], [9]. Insufficient potassium triggers flower and fruit drop, brown spots on leaves, and reduces resistance to drought and disease, drastically reducing fruit quality and quantity [1], [8]. Meanwhile, calcium deficiency results in plant tissue damage, disturbances in root growth, and the emergence of blossom end rot problems in fruit, which directly impacts the reduction of harvest quality [1]. Proper handling through balanced fertilization becomes the key to preventing this problem.

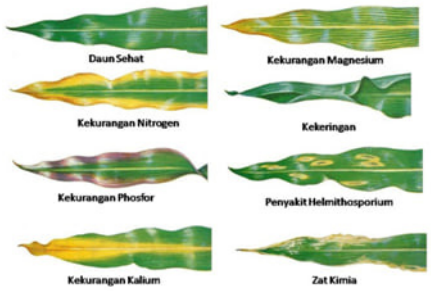

Figure 2. Examples of Macronutrient Deficiencies in Corn Plants

(Source: https://hortiindonesia.com)

3.2 Melon (*Cucumis melo*)

Melon (Cucumis melo) is a fruit that contains various vitamins and minerals beneficial for health. Cantaloupe melon is rich in vitamin C, vitamin A, potassium, vitamin B6, folic acid, and niacin. Vitamin A and vitamin C in cantaloupe contribute 54% and 49% of daily nutritional needs, respectively. Additionally, melon also contains various minerals such as potassium, calcium, iron, magnesium, phosphorus, sodium, and zinc. Orange-fleshed melon indicates the presence of carotenoids that are good for heart health and immune system, while green-fleshed melon contains vitamin B6 which is useful for maintaining bone and dental health [10].

Nutrition plays a very important role in melon cultivation because it directly affects growth, productivity, and harvest quality. Melon requires macronutrients such as nitrogen (N), calcium (Ca), and potassium (K) in optimal amounts to support plant physiological processes. Nitrogen is needed for chlorophyll and protein formation, which supports plant vegetative growth. Potassium helps improve fruit quality by strengthening plant tissues and supporting photosynthesis processes, while calcium plays a role in forming strong cell walls and preventing physiological damage to fruit. Nutrient deficiencies can cause various problems such as stunted growth, yellowing leaves, and defective fruit, which ultimately reduces the economic value of melon. Therefore, proper nutrient management is the key to achieving maximum and high-quality harvests.

Figure 3. Examples of Healthy Melon Plants

(Source: https://agri.kompas.com/)

3.3 Digital Image Processing

Digital image processing is a technological field that involves the application of computer-based algorithms to analyze and manipulate digital images in order to enhance visual quality and extract meaningful information [4]. This field encompasses various techniques, including contrast enhancement, noise reduction, and feature detection, which are widely used to support diverse applications such as plant disease diagnosis.

In the agricultural domain, digital imagery plays a crucial role in enabling the automatic analysis of plant symptoms. Image processing and computer vision algorithms can identify disease indicators by analyzing visual patterns, including changes in leaf color and texture, thereby supporting automated plant health assessment and early disease detection [7].

This technology involves several processing stages, including image segmentation, which aims to separate infected regions from the entire image to enable more accurate diagnosis. Feature extraction in digital image processing is used to identify and quantify visual characteristics such as edges, textures, and regions that are essential for classification tasks [11]. By applying these techniques, plant symptom analysis can be performed more efficiently and accurately, thereby supporting better agricultural management decisions.

3.4 Deep Learning

Deep Learning is one of the branches of Machine Learning (ML) that emphasizes how computers imitate the function of artificial neural networks (ANN) found in the human brain [2]. When the brain receives new information, it compares it with existing knowledge before understanding it. This understanding process involves recognizing new incoming things by utilizing previously recognized information. The brain labels each information according to certain categories, which can be very numerous and technical. For example, when we learn that an object is a ball, other similar objects such as basketballs or baseballs will also be recognized by the brain as balls thanks to their shape similarity. The "Deep" concept in deep learning imitates this process. This technique allows models to learn independently through several levels of layers in neural networks that can be used in various applications such as image recognition, natural language processing, or voice recognition [12].

Deep Learning is divided into three main stages named according to their layers, namely input layer, hidden layer, and output layer. Deep Learning models usually depend on the size of their hidden layer; some have few hidden layers (shallow hidden layer) and some are very complex so they are difficult to describe because of the very large number of hidden layers. Although hidden layer entities vary, Deep Learning is very efficient for handling problems with large volumes of unstructured data, especially when high accuracy results are desired [13].

3.5 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) is a development of Multilayer Perceptron (MLP) specifically designed to process data in two-dimensional format, such as images. As part of Deep Neural Network, CNN has a deeper network architecture and is often applied in visual data processing. Unlike MLP which tends to ignore spatial information because it considers each pixel as an independent feature, CNN is able to maintain spatial information, thus producing better performance in image classification tasks [3].

CNN concept was first introduced by Kunihiko Fukushima through a model known as NeoCognitron, which was designed for pattern recognition independent of object position changes [14]. This concept was then refined by Yann LeCun who developed the LeNet model for number and handwriting recognition [5]. CNN's success was increasingly recognized after Alex Krizhevsky won the ImageNet Large Scale Visual Recognition Challenge 2012 with the CNN architecture he developed, confirming the effectiveness of this method compared to other machine learning algorithms such as Support Vector Machine (SVM) [15].

3.5.1 CNN Architecture

Artificial Neural Networks (ANN) consist of various layers and a number of neurons in each layer. The determination of the number of layers and neurons has no fixed rules and tends to depend on the characteristics of the dataset used [16].

In the case of Multilayer Perceptron (MLP), networks without hidden layers are able to map linear equations, while networks with one or two hidden layers can map most equations on relatively simple data. However, on more complex data, MLP capabilities are limited. For networks with fewer than three hidden layers, there are approximation methods that can be used to determine the number of neurons in each layer to approach optimal results. Using more than two layers is often not recommended because it risks causing overfitting and significantly weakening the effectiveness of the backpropagation algorithm [16].

With the advancement of deep learning, it was found that MLP's shortcomings in handling complex data can be overcome by adding transformation functions that convert input data into a form that is easier to process by MLP. This triggered the development of deep

learning models that have several layers to transform data before classification. As a result, neural network models with more than three layers began to be developed. However, because the initial layers function as feature extraction methods, the number of layers in a deep neural network (DNN) has no universal rules and is highly dependent on the dataset used [16].

As a result, the number of layers and the number of neurons in each layer are considered as hyperparameters that need to be optimized through certain search methods.

On CNN (Convolutional Neural Network), the architecture consists of several main layers. Referring to the LeNet-5 architecture [17], there are four main types of layers in CNN. However, for this research, only three main types of layers are used:

Convolution Layer: The Convolution Layer in CNN is tasked with performing convolution operations on the output from the previous layer, which becomes the core process in CNN. Mathematically, convolution means applying a function to the output of another function repeatedly. In the context of image processing, convolution refers to the application of a kernel to all parts of the input image, with the kernel moving from the top left corner to the bottom right. Convolution is performed on image data to extract features from the input. This operation produces linear transformation on input data based on spatial information present in the image. Weights in this layer determine the convolution kernel used, so that the kernel can be trained according to the input received by CNN.

Subsampling Layer: Subsampling is a process used to reduce image data size. In image processing, subsampling also aims to improve feature position invariance, so features remain detected even if they experience position changes. One of the most commonly used subsampling methods in CNN is max pooling. In max pooling, the output from the convolution layer is divided into several small grids. Then, the maximum value from each grid is taken to form an image matrix with reduced size. This process ensures that detected features remain consistent, even if translation (shifting) occurs on objects in the image.

According to Springenberg, Dosovitskiy, Brox, and Riedmiller [6] the main function of pooling layer in CNN is to reduce image size. Therefore, pooling layers can be replaced by convolution layers with the same stride, without affecting the network results.

Fully Connected Layer: This layer, commonly used in MLP applications, aims to transform data dimensions so that data can be classified linearly. Each neuron in the convolution layer needs to be converted into one-dimensional data before it can be processed by the fully connected layer. However, this process causes loss of spatial information in the data and cannot be reversed. Therefore, fully connected layer is only applied at the end of the network.

4. Result and Dicussion

Implementation of the Convolutional Neural Network (CNN) model for classification of nutrient deficiencies in melon leaves was carried out through several stages, from data preparation, data understanding, data preparation, model creation, to evaluation and testing. The following is an explanation of each implementation step:

4.1. Data Preparation

The dataset used in this study was taken from the Kaggle platform. To be able to download the dataset, credentials in the form of a kaggle.json file were uploaded to Google Colab to provide access to the Kaggle API. The dataset was downloaded, extracted, and prepared for analysis purposes. The dataset consists of four classes: healthy, calcium deficiency, nitrogen deficiency, and potassium deficiency.

```
# Upload kaggle.json
from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}"'.format(
      name=fn))
```

```
 # Ubah permission file
!chmod 600 /content/kaggle.json

# Setup Kaggle environment
import os
os.environ['KAGGLE_CONFIG_DIR'] = "/content"

# Download dataset
!kaggle datasets download -d binnassor89/melon-
macronutrient-deficiency-dataset

# melakukan ekstraksi pada file zip
import zipfile
local_zip = 'melon-macronutrient-deficiency-dataset.zip'
zip_ref = zipfile.ZipFile(local_zip, 'r')
zip_ref.extractall('/content/melon-macronutrient-
deficiency-dataset')
zip_ref.close()
```

4.2. Data Understanding Result

Data understanding was carried out by calculating the number of images for each class and performing data visualization. There are a total of 200 images with balanced distribution in each class. Dataset visualization helps understand visual patterns and characteristics of each class. In addition, several sample images from each class are displayed to provide a general overview of the dataset.

```
# Mendefinisikan direktori utama dataset
base_dir = '/content/melon-macronutrient-deficiency-
dataset/Melon macronutrient deficeincy datasets'
print(os.listdir(base_dir))

# Menghitung jumlah gambar pada dataset
number_label = {}
total_files = 0
for i in os.listdir(base_dir):
    counting = len(os.listdir(os.path.join(base_dir, i)))
    number_label[i] = counting
    total_files += counting

print("Total Files : " + str(total_files))

# Visualisasi jumlah gambar tiap kelas
import matplotlib.pyplot as plt

plt.bar(number_label.keys(), number_label.values());
plt.title("Jumlah Gambar Tiap Label");
plt.xlabel('Label');
plt.ylabel('Jumlah Gambar');

# Menampilkan sampel gambar tiap kelas
import matplotlib.image as mpimg

img_each_class = 1
img_samples = {}
classes = list(number_label.keys())
```

```
for c in classes:
    temp = os.listdir(os.path.join(base_dir,
c))[:img_each_class]
    for item in temp:
        img_path = os.path.join(base_dir, c, item)
        img_samples[c] = img_path

for i in img_samples:
    fig = plt.gcf()
    img = mpimg.imread(img_samples[i])
    plt.title(i)
    plt.imshow(img)
    plt.show()
```

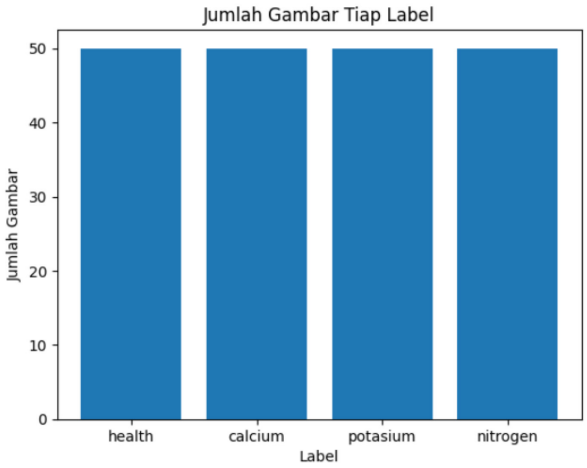

Figure 4. Graph of Data Amount for Each Label

4.3 Data Preparation Result

Data was further processed by dividing the dataset into two subsets, namely training data and validation data. This division was carried out with a ratio of 80:20 using ImageDataGenerator from TensorFlow. In addition, image augmentation techniques were applied to increase dataset variation with transformations such as flipping, rotation, zooming, and rescaling. This augmentation technique aims to make the model more robust to real data variations.

```
IMAGE_SIZE = (200,200)
BATCH_SIZE = 32
SEED = 999

# Menggunakan ImageDataGenerator untuk preprocessing
import tensorflow as tf

datagen =
tf.keras.preprocessing.image.ImageDataGenerator(
    validation_split=0.2
)

# Menyiapkan data train dan data validation
train_data = datagen.flow_from_directory(
    base_dir,
    class_mode='categorical',
    subset='training',
```

```
    target_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
    seed=SEED
)

valid_data = datagen.flow_from_directory(
    base_dir,
    class_mode='categorical',
    subset='validation',
    target_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
    seed=SEED
)

# Image Augmentation
data_augmentation = tf.keras.Sequential(
  [
    tf.keras.layers.RandomFlip("horizontal",
                      input_shape=(IMAGE_SIZE[0],
                                   IMAGE_SIZE[1],
                                   3)),
    tf.keras.layers.RandomRotation(0.1),
    tf.keras.layers.RandomZoom(0.1),
    tf.keras.layers.Rescaling(1./255)
  ]
)
```

4.4 CNN Modeling Result

The CNN model architecture was designed to handle image classification tasks with four classes. The model consists of several main layers, namely convolutional layers for feature extraction, pooling layers to reduce feature dimensions, dropout layers to prevent overfitting, and fully connected layers to produce final output. This model uses ReLU activation function in hidden layers and softmax in the output layer for multiclass classification.

The model was compiled with Adam optimization, categorical crossentropy loss function, and accuracy evaluation metric. After compilation, the model was trained using training data for 10 epochs with validation data as performance measurement. The training results show accuracy and loss on training and validation data visualized in graphs. The CNN model successfully achieved good accuracy with stable performance on validation data, indicating that the model is not overfitting and is able to generalize well.

```
# Membuat arsitektur model CNN
cnn_model = tf.keras.models.Sequential([
  data_augmentation,
  tf.keras.layers.Conv2D(32, 3, padding='same',
activation='relu'),
  tf.keras.layers.MaxPooling2D(),
  tf.keras.layers.Conv2D(64, 3, padding='same',
activation='relu'),
  tf.keras.layers.MaxPooling2D(),
  tf.keras.layers.Conv2D(64, 3, padding='same',
activation='relu'),
  tf.keras.layers.MaxPooling2D(),
  tf.keras.layers.Dropout(0.3),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(64, activation='relu'),
  tf.keras.layers.Dense(64, activation='relu'),
  tf.keras.layers.Dense(4, activation='softmax')
])
```

```
# Compiling model
cnn_model.compile(
    loss='categorical_crossentropy',
    optimizer=tf.keras.optimizers.Adam(),
    metrics=['accuracy']
  )

 # Training model CNN
cnn_hist = cnn_model.fit(
    train_data,
    epochs=10,
    validation_data = valid_data
)

acc = cnn_hist.history['accuracy']
val_acc = cnn_hist.history['val_accuracy']

loss = cnn_hist.history['loss']
val_loss = cnn_hist.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc)
plt.plot(epochs, val_acc)
plt.title('Training and validation accuracy')

plt.figure()

plt.plot(epochs, loss)
plt.plot(epochs, val_loss)
plt.title('Training and validation loss')
```

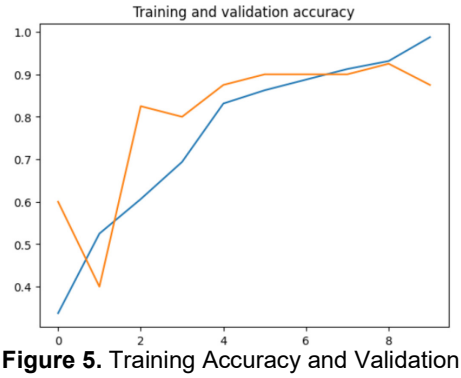

Figure 5. Training Accuracy and Validation

Figure 6. Training and Validation Loss

4.5 Model Evaluation Result

Additional evaluation was conducted with a confusion matrix to understand model performance in classifying each class. The results show that the model is able to classify melon leaf images with high accuracy in all four classes. The CNN model demonstrates good ability in capturing visual patterns, such as color and texture changes in leaves, which are indicators of nutrient deficiencies.

```
# Membuat plot akurasi model CNN
plt.figure(figsize=(10,4))
plt.plot(cnn_hist.history['accuracy'])
plt.plot(cnn_hist.history['val_accuracy'])
plt.title('CNN model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.grid(True)
plt.show()

print()

# Membuat plot loss model CNN
plt.figure(figsize=(10,4))
plt.plot(cnn_hist.history['loss'])
plt.plot(cnn_hist.history['val_loss'])
plt.title('CNN model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.grid(True)
plt.show()
```

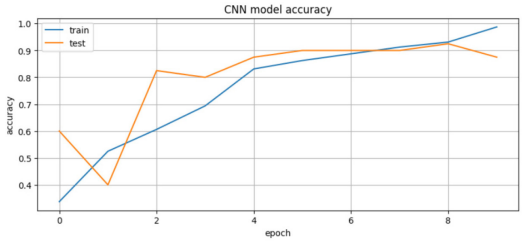

Figure 7. CNN Model Accuracy

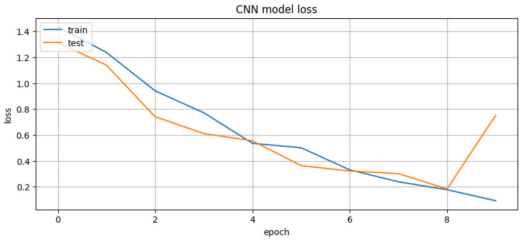

Figure 8. CNN Loss Model

4.6 Model Testing

The trained CNN model was tested using new images to evaluate its performance in real conditions. In testing, the model successfully identified melon leaf images into four categories: Healthy, Calcium Deficiency, Nitrogen Deficiency, and Potassium Deficiency. Testing was carried out by uploading a new image, processing the image through the preprocessing pipeline, and predicting the class using the model. This process shows that the model can automatically detect melon leaf conditions and provide prediction results that match the actual image conditions.

```
# Menampilkan daftar kelas atau label gambar
train_data.class_indices

# Menguji coba model
import numpy as np
from keras.preprocessing import image
%matplotlib inline

uploaded = files.upload()

for fn in uploaded.keys():

  # predicting images
  path = fn
  img = tf.keras.utils.load_img(path,
target_size=IMAGE_SIZE)
  imgplot = plt.imshow(img)
  x = tf.keras.utils.img_to_array(img)
  x = np.expand_dims(x, axis=0)

  images = np.vstack([x])
  classes = cnn_model.predict(images,
batch_size=BATCH_SIZE)
  classes = np.argmax(classes)

  print(fn)
  if classes==0:
    print('Sehat')
  elif classes==1:
    print('Kekurangan Kalsium')
  elif classes==2:
    print('Kekurangan Nitrogen')
  elif classes==3:
    print('Kekurangan Potasium')
```

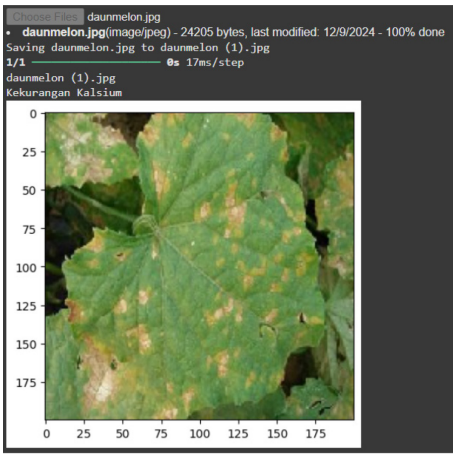

Figure 9. CNN Model Test Results

4.7 Deployment

The trained model was prepared for use in further implementation. The model was exported to HDF5 format for desktop-based applications and to TFLite format for

implementation on mobile or IoT devices. With this, the model can be used directly by users to detect nutrient deficiencies in melon leaves through camera-based systems.

```
#Model format HDF5 yang digunakan untuk mobile
cnn_model.save('model-melon-leaf-recognition.h5')

#Model format TFLite yang digunakan untuk website
converter =
tf.lite.TFLiteConverter.from_keras_model(cnn_model)
tflite_model = converter.convert()
with tf.io.gfile.GFile('model-melon-leaf-recognition.h5',
'wb') as f:
  f.write(tflite_model)
```

5. Conclusion

This study successfully developed a classification model based on Convolutional Neural Network to detect macronutrient deficiencies in melon leaves, namely calcium, nitrogen, and potassium deficiencies, as well as detecting healthy leaves. The model was trained using a dataset consisting of 200 leaf images with balanced distribution in each class. The training and evaluation process showed accurate results with stable performance on validation data.

Through implementation of the CRISP-DM methodology, this research covers all stages from business needs understanding, data exploration and preparation, model building, to model evaluation and implementation. The data augmentation process, such as flipping, rotation, and zooming, helps improve model robustness to real image variations. The CNN model demonstrates good ability in capturing visual patterns, such as color and texture changes in leaves, which are indicators of nutrient deficiencies.

Testing results show that the model can recognize new melon leaf images with high accuracy, classifying them into the four predetermined categories. This model has also been successfully implemented in HDF5 format for desktop-based applications and TFLite for mobile devices, making it ready for use by farmers as a tool for early detection of nutrient deficiencies. This research contributes to agricultural science by providing an automated, accurate, and efficient solution for diagnosing plant nutritional problems, enabling farmers to take timely preventive actions to improve crop yield and quality.

References

[1] D. R. Nurhayati, *Pengantar Nutrisi Tanaman*. Surakarta, Indonesia: UNISRI Press, 2021.

[2] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, “A Survey on Deep Transfer Learning,” in *Proceedings of the International Conference on Artificial Neural Networks*, Springer, 2018, pp. 270–279.

[3] W. S. E. Putra, “Klasifikasi Citra Menggunakan Convolutional Neural Network ({CNN}) pada {Caltech} 101,” *J. Tek. ITS*, vol. 5, no. 1, 2016.

[4] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 4th ed. New York, NY, USA: Pearson Education, 2018.

[5] Y. LeCun *et al.*, “Handwritten Digit Recognition with a Back-Propagation Network,” in *Advances in Neural Information Processing Systems*, 1990, pp. 396–404.

[6] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, “Striving for Simplicity: The All Convolutional Net,” in *Proceedings of the International Conference on Learning Representations ({ICLR})*, 2015.

[7] R. Szeliski, *Computer Vision: Algorithms and Applications*, 2nd ed. Cham, Switzerland:

Springer, 2020.

[8] Fakultas Pertanian UMSU, “Peran Penting Unsur Hara Makro,” 2023.

[9] T. Harjoko, “Peran Nitrogen pada Tanaman,” Malang, Indonesia, 2005.

[10] U.S. Department of Agriculture, “National Nutrient Database for Standard Reference Release 28,” 2016.

[11] R. C. Gonzalez, R. E. Woods, and S. L. Eddins, *Digital Image Processing Using {MATLAB}*, 3rd ed. New York, NY, USA: Pearson Education, 2020.

[12] H. A. Dau and N. Salim, “Introduction to Deep Learning: A Focus on Visual Data Processing and Natural Language Understanding,” *J. Data Sci.*, vol. 18, no. 4, pp. 215–229, 2020.

[13] A. K. Jakhar and A. Kaur, “Machine Learning and Its Applications: A Comprehensive Study,” *J. Comput. Sci. Technol.*, vol. 35, no. 4, pp. 521–534, 2020.

[14] K. Fukushima, “Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position,” *Biol. Cybern.*, vol. 36, no. 4, pp. 193–202, 1980.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “{ImageNet} Classification with Deep Convolutional Neural Networks,” in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[16] D. Stathakis, “How Many Hidden Layers and Nodes?,” *Int. J. Remote Sens.*, vol. 29, no. 8, pp. 2133–2147, 2008.

[17] Stanford University, “An Introduction to Convolutional Neural Networks,” 2023.