Detection of Coffee Leaf Diseases Based on Image Features Using Convolutional Neural Networks with pre-train VGG16 Model

Pradita Eko Prasetyo Utomo[*1)], Benedika Ferdian Hutabarat [1)], Niken Rarasati [2)], Fintria Lairatulrahmi. SY [1)] , Muhammad Sadam Abdillah Hawid [1)]

*Corresponding author
*ORCHID IDs: https://orcid.org/0009-0008-4787-4609

[1)]Department of Information System, Universitas Jambi, Indonesia
[2)] Department of Mathematics, Universitas Jambi, Indonesia

*email: pradita.eko@unja.ac.id

Abstract

Leaf diseases in coffee plants can reduce plant productivity and reduce crop yields. This leaf disease is caused by various types of fungi and bacteria that attack the leaves of coffee plants. Control of leaf diseases in coffee plants can be done by regular observation and the use of pesticides. However, the use of pesticides can have negative impacts on the environment and human health. Therefore, there is a need for technology that can help farmers identify leaf diseases on coffee plants quickly and accurately. A Technology that can be used is image processing using deep learning techniques. Image processing using deep learning techniques has been widely used in various applications. Convolutional Neural Network is a type of deep learning that is often used in image processing. This research proposes the detection of coffee leaf diseases by using images of coffee leaves to distinguish between normal and diseased coffee leaves. For detection modelling, the proposed pre-trained model VGG16 is used. This model was trained using transfer learning techniques on the training set and produced 93.2% accuracy on the test set. This model is expected to help farmers identify and overcome disease problems in coffee plants more quickly and effectively.

Keyword: CNN, VGG16, leaf of coffee, diseases, deep learning

1. Background

The success rate of harvesting coffee plants is influenced by the quality of the coffee plants produced. Many factors influence the success of a coffee plant harvest, one of the main ones is plant productivity, which can be seen from the leaves. Leaves are an important part because the photosynthesis process occurs in the leaves, so leaf health needs to be considered.

Leaf diseases on coffee plants are a serious problem for coffee farmers in Indonesia. Leaf diseases in coffee plants can reduce plant productivity and reduce crop yields. This leaf disease is caused by various types of fungi and bacteria that attack the leaves of coffee plants, such as Hemileia vastatrix, Cercospora coffeicola, and Phoma sp. (Agrios, 2005). Control of leaf diseases in coffee plants can be done by regular observation and the use of pesticides. However, the use of pesticides can have negative impacts on the environment and human health. Therefore, there is a need for technology that can help farmers identify leaf diseases on coffee plants quickly and accurately. One technology that can be used is image processing using deep learning techniques. Image processing using deep learning techniques has been widely used in various applications, such as face recognition, object detection, and natural language recognition (Goodfellow et al., 2016).

Convolutional Neural Network (CNN) is a type of deep learning that is often used in image processing. Research on the classification of leaf diseases in coffee plants using deep learning technology and CNN has been carried out previously. For example, Irfansyah et.al (2021) conducted research implementing the Alexnet Convolutional Neural Network (CNN) architecture with the MATLAB programming platform for disease identification in coffee plants through images. The training process involving 260 training data produced an accuracy value of 69.44-80.56%. The network testing process using 40 test data resulted in an accuracy of 81.6%. However, this research uses a relatively complex CNN architecture and requires large computing resources. Therefore, this study will use the lighter and more resource-efficient VGG16 architecture to classify leaf diseases in coffee plants. This research aims to implement the Convolutional Neural Network algorithm with the VGG16 architecture to form a disease classification model in coffee plants. It is hoped that this classification model can help farmers identify diseases in coffee plants quickly and accurately, so that they can increase plant productivity and crop yields.

2. Research Methods

This research will be carried out at the ICT Laboratory of the Faculty of Science and Technology, Jambi University, which is located at the Pinang Masak Campus, Jambi University, Jalan Raya Jambi Muara Bulian KM 15, Mendalo Indah, Muaro Jambi and data collection will be carried out at the Department of Agriculture and Horticulture, Jambi Province. This research began to be carried out since the proposal was approved.

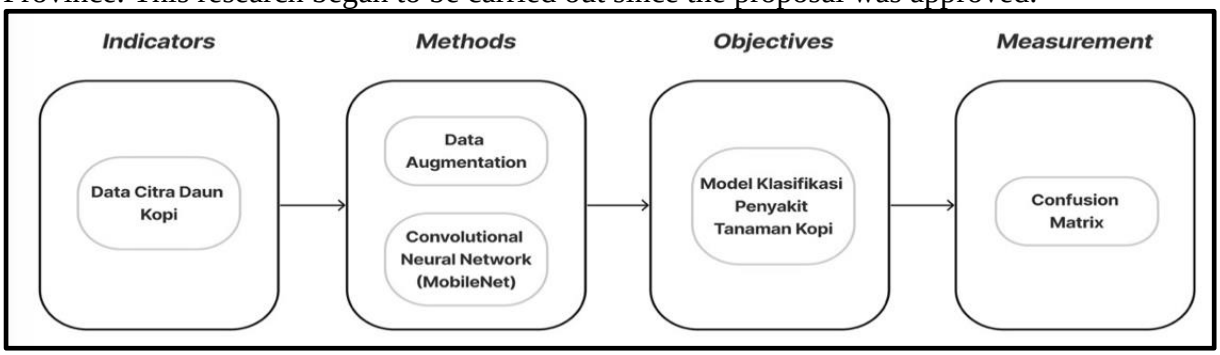


Figure 1. Research framework

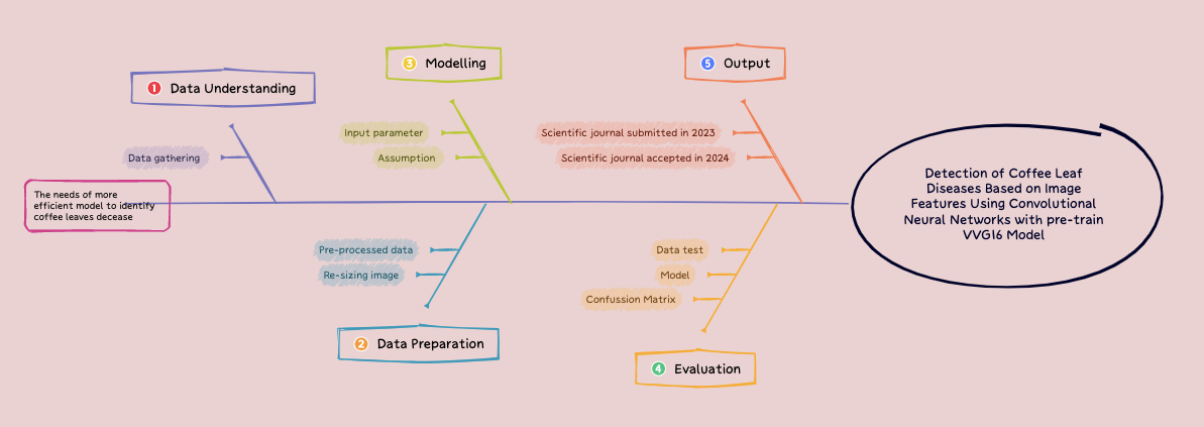


Figure 2. *Fishbone Diagram*

Figure 2 explains the stages of this research, the details of the research stages are explained as follows:

1. Data Understanding

In this first stage, a dataset will be collected where the dataset is obtained from Kaggle. An example of the data that will be used is in Figure 3.

Pada Gambar 2 menjelaskan tentang tahapan dari penelitian ini, adapun detail tahapan penelitian dijelaskan sebagai berikut:

(a) (b) (c)

Figure 3. Example of Coffee Leaf Image Data (a) healthy leaves (b) leaves with red spider mites (c)

1. Data Preparation/Data Preprocessing

This stage (which can be seen in Figure 4) includes dividing the dataset to be used and cleaning the data. Then the Data Augmentation stage will also be carried out.

Figure 4. Flow of data preparation

2. Data Augmentation

Augmentation is a way to overcome overfitting by modifying data into various shapes using techniques such as rotation, zoom, horizontal flip, width shift, height shift, and so on. The things that can be done at the data argumentation stage are as shown in Figure 5 below:

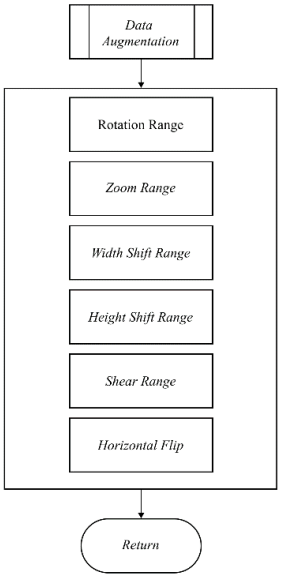


Figure 5. Flow of data *Augmentation*

The more there are and are varied, the more influence it will have on machine learning during training. The more data the more accuracy results in the training process can improve. The problem that arises when there is too little data is overfitting.

3. Modelling

At this stage modeling will be carried out to obtain a classification model. The method applied is a classification method in deep learning with the Convolutional Neural Network algorithm which is carried out using the VGG16 architecture. The VGG16 architecture was trained and tested using the Python language with the Tensorflow CPU library. The VGG16 architecture can be seen in Figure 6. In this research, 8 layers will be used which are designed for coffee leaf image classification and each image is used many times in the training process. During model training, the learning algorithm will experience each training batch exactly once during the epoch, and will be assessed for performance at the end of the epoch on each validation set.

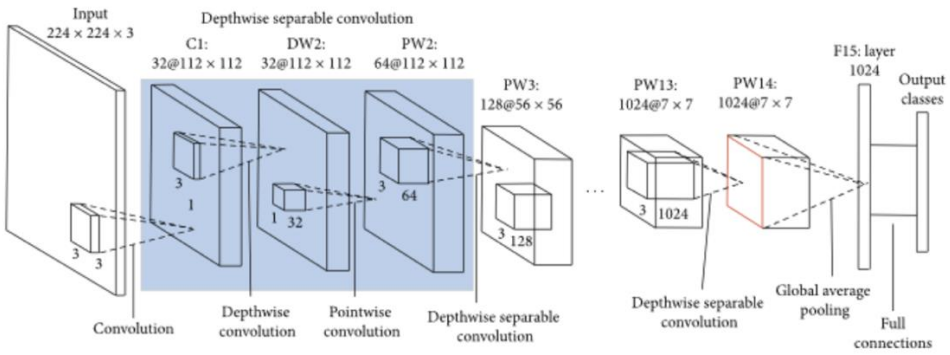


Figure 6. Architecture of VGG16

3. Evaluation

This evaluation stage is carried out to measure whether the model created is good for use or not. This stage uses the Confusion Matrix as a reference. So we can make a decision whether to make another model or not. Figure 7 is the evaluation stages:

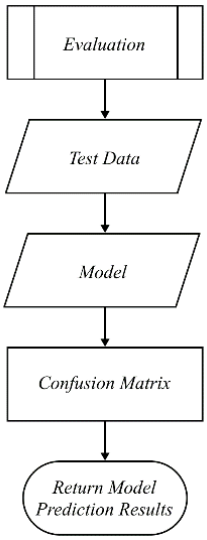


Figure 7. Flow of *evaluation*

3. Results and Discussion

1. Preprocessing Data

The data used in this research consists of secondary and primary datasets. The secondary dataset is a dataset that comes from Kaggle where the amount of data is 1,657 data consisting of 284 normal leaves, 332 miner leaf diseases, 388 phoma leaf diseases, 393 red spider mite leaf diseases, and 260 rust leaf diseases. Meanwhile, the primary datasets were taken directly from the BPP (Agricultural Extension Center) and coffee land/plantations located in Dendang District, East Tanjung Jabung Regency. This research consists of stages, namely preprocessing, feature extraction, and identification stage.

At the data processing stage, the background will be removed using the rembg library. This rembg itself is used with the aim of removing the background from an image so that it can produce an image with a transparent background so that it can focus on the main object of the image and the modeling process can work optimally in reading the image pixels. The preprocessing process for removing the background can be seen in pictures 8, 9 and 10.

```
def remove_background(input_folder, output_folder):
    # Buat folder output jika belum ada
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    # Loop melalui setiap file gambar dalam folder input, hapus background, dan simpan hasilnya di folder output
    file_list = os.listdir(input_folder)

    for filename in file_list:
        input_filepath = os.path.join(input_folder, filename)
        output_filepath = os.path.join(output_folder, filename)

        # Membuka file gambar
        with open(input_filepath, "rb") as img_file:
            # Menghapus latar belakang menggunakan Rembg
            output_img = rembg.remove(img_file.read())

        # Menyimpan gambar tanpa latar belakang di folder output
        with open(output_filepath, "wb") as output_file:
            output_file.write(output_img)
```

```
# Tentukan path folder input dan output untuk kedua folder
input_folder1 = '/content/drive/MyDrive/kopi/miner'
output_folder1 = '/content/drive/MyDrive/rembg2/miner'

input_folder2 = '/content/drive/MyDrive/kopi/nodisease'
output_folder2 = '/content/drive/MyDrive/rembg2/nodisease'

input_folder3 = '/content/drive/MyDrive/kopi/phoma'
output_folder3 = '/content/drive/MyDrive/rembg2/phoma'

input_folder4 = '/content/drive/MyDrive/kopi/Red_spider_mite'
output_folder4 = '/content/drive/MyDrive/rembg2/Red_spider_mite'

input_folder5 = '/content/drive/MyDrive/kopi/rust'
output_folder5 = '/content/drive/MyDrive/rembg2/rust'

# Hapus background dari folder pertama
remove_background(input_folder1, output_folder1)

# Hapus background dari folder kedua
remove_background(input_folder2, output_folder2)

# Hapus background dari folder kedua
remove_background(input_folder3, output_folder3)

# Hapus background dari folder kedua
remove_background(input_folder4, output_folder4)
```

Figure 8. background removal processes

Below is the dataset before background removal

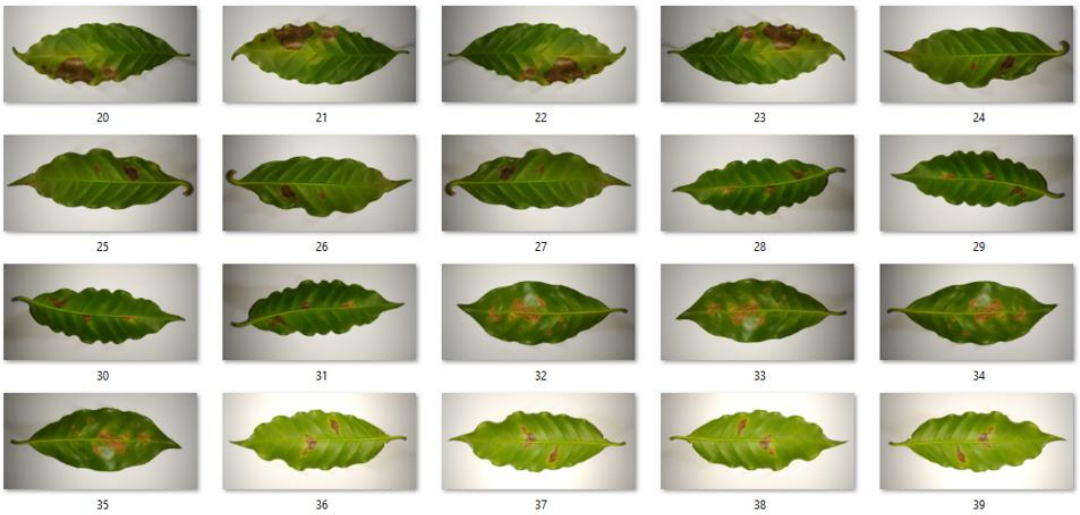


Figure 9. dataset before background removal

The following are the results of removing the background using rembg

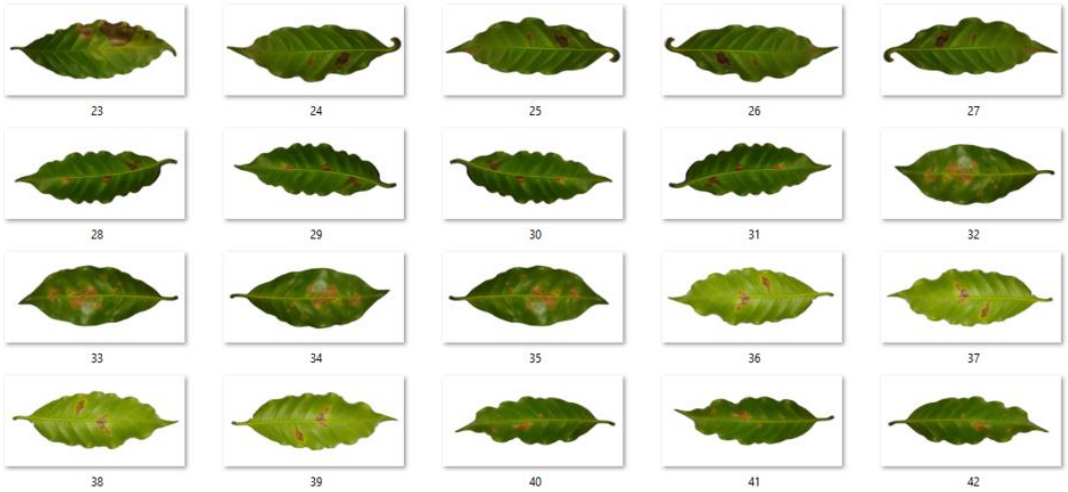


Figure 10. Result of background removal

From the image above it can be seen that the results of removing the background using the rembg library are very good where the image background becomes transparent so that the machine can focus on reading the main object and modeling can be done optimally.

Later on, we initiate the VGG16 architecture by using the following language as shown in figure 11.

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import backend as K
from tensorflow.keras.layers import Dense, Activation,Dropout,Conv2D, Ma
from tensorflow.keras.optimizers import Adam, Adamax
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras import regularizers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Model, load_model, Sequential
from keras.callbacks import ModelCheckpoint
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from matplotlib.pyplot import imshow
import os
import seaborn as sns
sns.set_style('darkgrid')
from sklearn.metrics import confusion_matrix, classification_report
from IPython.display import display, HTML
```

Figure 11. Initiate the VGG16 architecture

Next, we initiate the image generator by using the following language as shown in figure 12.

```
from keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(
        rotation_range=40,
        width_shift_range=0.2,
        height_shift_range=0.2,
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest')
```

Figure 12. Initiate the Image generator.

And also determine the directory for the output data as shown in figure 13.

```
import os

save_dir = 'D:\datasets\augmentasi'  # Nama direktori yang Anda inginkar
```

Figure 13. Determine the output directory

After that, the next step is split data. In this process, the data is divided, including 70% for train data, 20% for test data, and 10% for valid data. This 70%-20%-10% ratio is commonly used. Using 70% of the train data itself can help the model to train the data so that the model can understand patterns in the data better. Data split command is shown in Figure 14.

```
train_split=.7
test_split=.2
dummy_split=test_split/(1-train_split)
train_df, dummy_df=train_test_split(df, train_size=train_split, shuffle=
test_df, valid_df=train_test_split(dummy_df, train_size=dummy_split, shu
print ('train_df length: ', len(train_df), '  test_df length: ', len(tes
```

Figure 14. Data split

The division of data into train, test, and valid data is used to measure model performance, besides this data division can help identify overfitting. By using this distribution of test data, we can monitor the work of the model and then if overfitting occurs, we can take action as soon as possible.

After creating the model, the next step is to carry out the training process, where the training process uses 10 epochs. One epoch is when all data passes through one time, forward and backward through all neural network nodes in one time. In this research, 10 iterations (looping) were carried out as shown in figure 15.

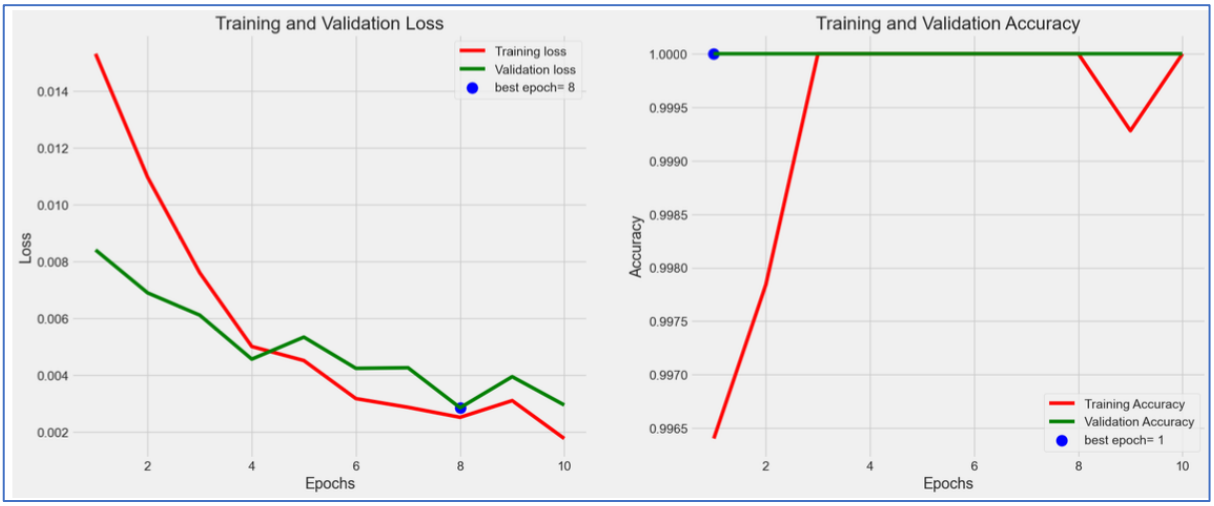
```
epochs =10

history=model.fit(x=train_gen, epochs=epochs, validation_data=valid_gen)
```

```
Epoch 1/10
22/22 [==============================] - 686s 31s/step - loss: 0.015
3 - accuracy: 0.9964 - val_loss: 0.0084 - val_accuracy: 1.0000
Epoch 2/10
22/22 [==============================] - 653s 30s/step - loss: 0.011
0 - accuracy: 0.9978 - val_loss: 0.0069 - val_accuracy: 1.0000
Epoch 3/10
22/22 [==============================] - 647s 30s/step - loss: 0.007
6 - accuracy: 1.0000 - val_loss: 0.0061 - val_accuracy: 1.0000
Epoch 4/10
22/22 [==============================] - 652s 30s/step - loss: 0.005
0 - accuracy: 1.0000 - val_loss: 0.0046 - val_accuracy: 1.0000
Epoch 5/10
22/22 [==============================] - 666s 30s/step - loss: 0.004
5 - accuracy: 1.0000 - val_loss: 0.0053 - val_accuracy: 1.0000
Epoch 6/10
22/22 [==============================] - 675s 31s/step - loss: 0.003
2 - accuracy: 1.0000 - val_loss: 0.0042 - val_accuracy: 1.0000
Epoch 7/10
```

Figure 15. Data training using 10 epoch

Result of the VGG16 can be seen in figure 16 bellow. Based on the graph we can see that the traing and validation loss showing a good result starting from epoch 8 and for training and validation accurasi, the model got 100% accuracy starting from epoch 3.



4. Conclusion

The classification model for coffee plant leaf diseases using the VGG16 Convolutional Neural Network (CNN) architecture has been proven to have very good accuracy of 100%. This success underscores the model's outstanding ability to precisely identify various diseases affecting coffee plant leaves. These results instill hope that the application of artificial intelligence technologies, such as CNNs, can serve as effective tools in early detection and management of plant diseases, ultimately enhancing the overall productivity and well-being of the coffee farming industry.

5. Reference

Agrios, G. N. (2005). Plant Pathology (5th ed.). Elsevier Academic Press.

Elfatimi, O., Karouach, R., & Sayouti, M. (2022). Beans Leaf Diseases Classification Using MobileNet Models. International Journal of Advanced Computer Science and Applications, 13(2), 234-242.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

Hasan, M. N., Islam, M. R., Islam, M. R., Islam, M. R., & Mahmud, S. (2021). Klasifikasi penyakit citra daun anggur menggunakan model CNN-VGG16. Jurnal Teknologi dan Sistem Komputer, 9(2), 100-106.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv preprint arXiv:1704.04861.

Irfansyah, M., Putra, A. S., & Fadlika, I. A. (2021). Arsitektur Convolutional Neural Network (CNN) Alexnet Untuk Klasifikasi Hama Pada Citra Daun Tanaman Kopi. Jurnal Informatika, 8(2), 131-140.

Lewis, K. J., Tan, Y. P., & Fook, V. F. (2020). Image Processing Techniques and Data Mining Algorithms for Coffee Plant's Leaves Classification. International Journal of Advanced Science and Technology, 29(7), 3193-3200.

Milosevic, N. (2020). Introduction to Convolutional Neural Networks. In Introduction to Convolutional Neural Networks. https://doi.org/10.1007/978-1-4842-5648-0

Nagamani, B. H., & Sarojadevi, H. (2022). Tomato Leaf Disease Detection using Deep Learning Techniques. International Journal of Computer Sciences and Engineering, 10(4), 298-305.

Sharma, A., Verma, S. K., & Mishra, S. K. (2022). Plant Disease Diagnosis and Image Classification Using Deep Learning. International Journal of Intelligent Computing and Cybernetics, 15(1), 66-80.

Zaki, A. M., Alqarni, A. S., Aljuaid, S. S., Alhassan, R. A., Alqahtani, A. S., & Alsolamy, A. O. (2020). Classification of tomato leaf diseases using MobileNet V2. IEEE Access, 8, 121431-121442.