Journal of Electrical Engineering and Computer (JEECOM)
Vol. 7, No. 1 (2025), DOI: 10.33650/jeecom.v4i2
p-ISSN: 2715-0410; e-ISSN: 2715-6427

Classification Of Mustard Leaf Diseases Using Convolutional Neural Network Architecture

M. Hafidurrohman [1], Kusrini [2]
[1,2] Magister Teknik Informatika
Universitas Amikom Yogyakarta, Indonesia

Article Info

Article history:

Received February 11, 2025
Revised February 14, 2025
Accepted April 10, 2025

Keywords:

Convolutional Neural Network
Xception
VGG19
Classification
Mustard Leaves

ABSTRACT

Diseases in mustard leaves can reduce productivity if not detected early. This study aims to develop and evaluate a disease classification system for mustard leaves using Convolutional Neural Network (CNN) architectures, specifically Xception and VGG19, while comparing their performance in terms of accuracy and computational efficiency. The mustard leaf image dataset undergoes preprocessing before being used for model training and testing. Experimental results show that Xception achieves the highest validation accuracy of 99% with better loss stability compared to VGG19, which attains 94.50% accuracy but exhibits greater fluctuation. In terms of time efficiency, VGG19 reaches optimal accuracy faster and completes the training process in 42 seconds, whereas Xception requires more epochs and a training time of 50 seconds. Therefore, Xception is recommended for classification tasks that demand high accuracy and stability, while VGG19 is more suitable for rapid detection with a slight trade-off in accuracy stability.

Corresponding Author:
M. Hafidurrohman,
Universitas Amikom Yogyakarta
Jl. Ring Road Utara, Ngringin, Condongcatur, Daerah Istimewa Yogyakarta 55281, Indonesia
Email : m_hafidurrohman04@students.amikom.ac.id

1. INTRODUCTION

Mustard leaf (*Brassica juncea*) is one of the popular green vegetables in many countries, especially in Asia. The plant is characterized by its wavy broad leaves with thick white stems and thrives in some areas with sufficient irrigation and dry characteristics[1]. Aside from being easy to cultivate, mustard leaves are rich in nutrients and vitamins, making them a nutritious vegetable option, while their seeds can also be utilized for mustard oil production[2]. As a vegetable with significant economic value, mustard leaves are widely consumed by the public. However, they are susceptible to various diseases that can damage the plants before harvest[3]. Research conducted by [4] indicates that diseases in mustard leaves generally affect the leaves and stems, with primary symptoms including leaf spots and discoloration, which serve as indicators of bacterial or fungal infections. Diseases affecting mustard leaves not only impact the quality and quantity of the harvest but also influence product availability in the market and farmers' income.

Early detection of diseases in mustard leaves is a crucial step in maintaining plant health and ensuring a high-quality harvest[5]. Manual observation using traditional methods is often time-consuming, requires specialized expertise, and carries a high risk of errors. Therefore, the application of technology is necessary to improve efficiency in detecting and managing plant diseases. Recent studies have leveraged advanced technologies such as *Convolutional Neural Network (CNN)* to effectively identify and classify diseases in

mustard leaves. CNN, which excel in image classification through complex visual features such as color, texture, and symptom shape, can distinguish between healthy and infected leaves[6]. Models such as Sequential CNN, ResNet-50, and VGG have demonstrated high accuracy in detecting mustard leaf diseases, providing farmers with an effective tool for early detection. This CNN-based approach helps prevent disease spread and enhances crop quality through more efficient management.

Research conducted by [7] also found that applying CNN algorithms for pest detection in mustard leaves yielded significant results, achieving an accuracy of up to 92%. This approach can help farmers optimize their harvest and prevent losses caused by pest infestations. Although CNN has proven to deliver high accuracy in classifying mustard leaf diseases, challenges such as overfitting, underfitting, and model architecture optimization remain obstacles[8]. The selection of architectures such as ResNet-50 and Xception has shown promising results, but hyperparameter exploration—such as the number of filters, kernel size, and dropout rate—remains limited. Additionally, techniques like data augmentation and optimization of training methods, such as Adam or SGD, are necessary to enhance model performance. Therefore, further development through parameter exploration, data augmentation, and the use of diverse datasets is crucial to supporting sustainable agriculture and improving the efficiency of plant disease detection[9].

In this context, this study aims to enhance the performance of disease classification in mustard leaves by optimizing parameters in the CNN model architectures, namely Xception and VGG19. The optimization process involves adjusting the learning rate, dropout rate, and the number of filters in each convolutional layer. The dataset used in this study is sourced from Kaggle and has been utilized in previous research. As an innovative approach, this study applies parameter optimization techniques to further improve model performance and conducts a comparative analysis of both CNN architectures. Additionally, this research evaluates the computational speed of both models to identify a balance between accuracy and processing efficiency. Thus, this study is expected to contribute to improving the accuracy of early disease detection in plants and providing insights into model processing efficiency, ultimately assisting farmers in making faster and more accurate decisions.

2. METHOD

This research includes a series of systematic steps reflected in the research model to be conducted, from data collection to conclusion. This section describes the research methods used to build and evaluate performance of the model, including the methods chosen to obtain the dataset, data preparation techniques, and data analysis techniques, as visualized in Figure 1.

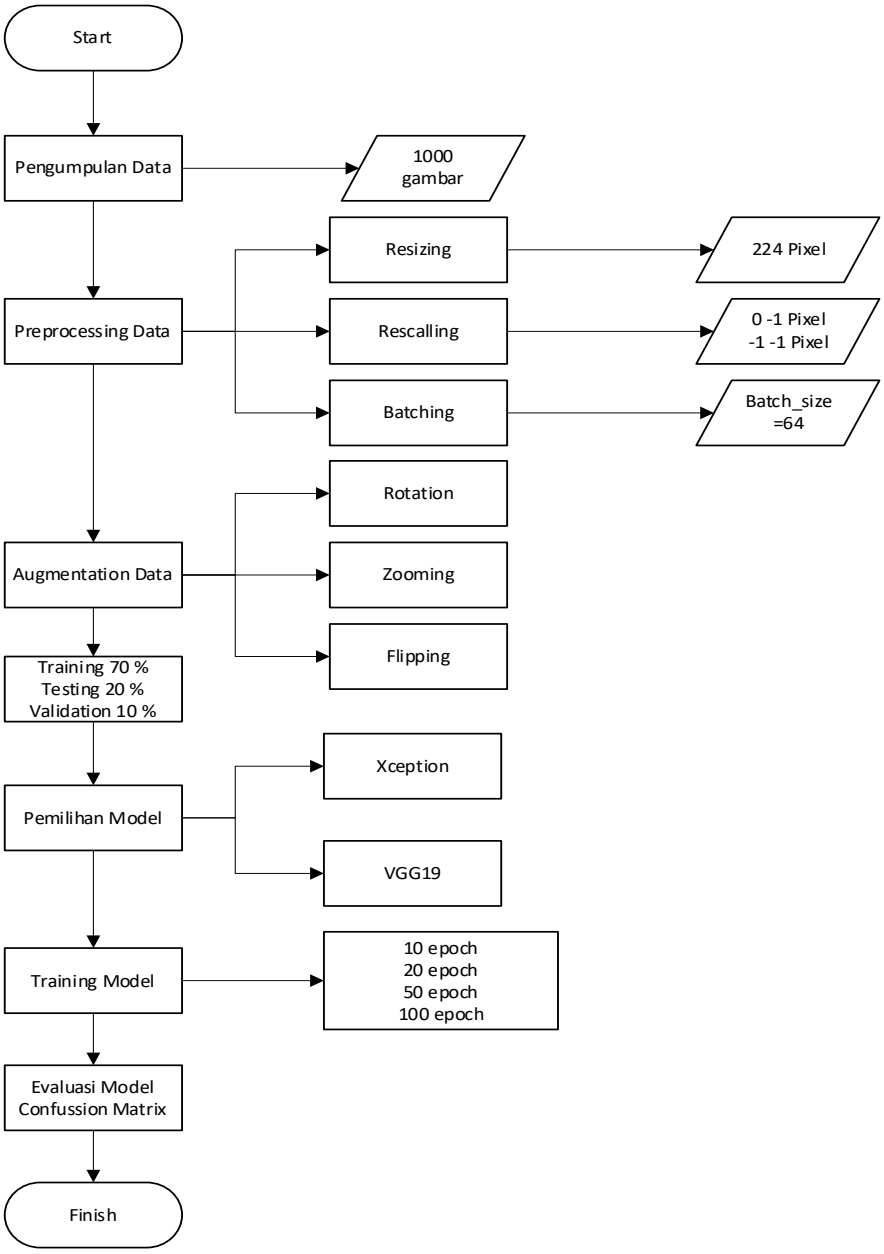

Figure 1. Research Stages

2.1 Literature Review

This research involves collecting data through analyzing previous research relevant to the topic of plant disease classification. The data used can be public data available in media publications, public repositories, or other sources, including archives that are allowed to be used publicly. Such data sources include repositories such as *GitHub, UCI Machine Learning, Kaggle*, and similar platforms that have been collected by previous researchers.

2.2 Dataset Collection

This research uses a dataset obtained from *Kaggle*, namely the *Caisim Dataset* [7] and consists of 2 disease classes with 1000 image data. This dataset consists of 500 images of *Daun Sawi Ada Hama* and 500 *Daun Sawi Tanpa Hama.* The following image shows a sample of the dataset that will be used.

Gambar 2. Dataset

2.3 Preprocessing data

This stage aims to ensure that the data used in the model is cleaner, well-structured, and ready for processing, thereby maximizing the classification results of mustard leaf diseases. This process includes handling imbalanced datasets, *resizing*, normalization (*rescaling*), and splitting the data into batches (*batching*)[10].

2.3.1 Resizing

This stage aims to change the size of the image so that it has the same or uniform dimensions[11]. For example, all images are converted to 224x24 pixels. This size consistency is very important, especially for CNNs, which require fixed-size inputs in order to apply operations such as convolution and pooling consistently.

2.3.2 Rescaling

This stage aims to convert the pixel values from a scale of 0-255 to a smaller range, for example from 0 to 1 or -1 to 1[11]. Since most images are usually stored with a pixel value of 255, but deep learning models often work more optimally when the input is smaller.

2.3.3 Batching

This stage has the purpose of dividing the dataset used into small groups or batches [12]. This method can help reduce memory requirements, which enables model training on devices with limited memory capacity, as the model does not need to hold the entire dataset in memory at once. The parameter used by the researcher is batch=64.

2.4 Augmentation data

This step is a process to increase the diversity of the training data without increasing the amount of original data, so that the model is more robust to input variations and avoids *overfitting* [13]. In image classification, this technique includes *rotation, zooming and flipping* to make the model more generalizable to new data.

2.5 Splitting Dataset

This stage is the process of dividing the prepared data into three main subsets, namely *training data*, *testing data*, and *validation data*. This division aims to allow the model to be trained and evaluated more accurately, as well as minimizing the risk of *overfitting* [14]. In this study, the *training data* is set at 70% of the data will be used to train the model. Then, the remaining 30% data will be divided back into *testing data* and *validation data*.

2.6 Model Search

After *preprocessing*, *data augmentation*, and *data splitting*, experiments were conducted on two selected models, VGG19 and Xception. In this process, various combinations of *learning rates*, *dropout*, and *filters* were tested to obtain the best results.

2.7 Model Evaluation

The model was evaluated using a confusion matrix to gain a deeper understanding of the prediction error the model made[15]. Each element in the confusion matrix indicates how many predictions are correct or incorrect for each class, thus providing a clear insight into where the model often misclassifies images.

3. RESULTS AND DISCUSSION

The testing in this research was conducted using a Lenovo V14 laptop with specifications of 11th generation Core i3 processor, 12GB RAM, 256GB SSD, and Windows 11 Pro operating system, while utilizing *Google Colab* as a supporting tool for program implementation. The results and discussion of this research will be outlined as follows :

3.1. Training Scenario

This study will compare the performance of Xception and VGG19 models for mustard plant classification with different variations of *learning rate (0.001, 0.0001), dropout rate (0.3, 0.5, 0.7), and number of filters (64, 32, 16).* This experiment was conducted using a dataset obtained from *Kaggle* with appropriate preprocessing. The main objective is to analyze the accuracy and computational speed of each *hyperparameter* combination to determine the best configuration to improve classification performance.

Table 1. Scenario

Eksperiment	Model	Learning Rate (LR)	Dropout Rate	Filter
1	Xception	0.001	0.3	64
2	Xception	0.001	0.5	32
3	Xception	0.001	0.7	16
4	Xception	0.0001	0.3	64
5	Xception	0.0001	0.5	32
6	VGG19	0.001	0.3	64
7	VGG19	0.001	0.5	32
8	VGG19	0.001	0.7	16
9	VGG19	0.0001	0.3	64
10	VGG19	0.0001	0.5	32

3.2. Result

Based on the scenarios conducted in Table 1, the Xception model showed the best performance in classifying mustard leaf diseases with a final accuracy of 99%, which was obtained using a *learning rate* of 0.0001, a *dropout* of 0.5, and a *filter* count of 32. Meanwhile, the VGG19 model achieved an accuracy of 94.50% with a *learning rate* of 0.001, a *dropout* of 0.5, and the same number of *filters*. Although Xception is superior in terms of accuracy, the VGG19 model is more efficient in training speed. VGG19 can complete the training and validation process in 42 seconds, while Xception takes 50 seconds.

```
Epoch 1/100
11/11 [==============================] - 134s 12s/step - loss: 0.4686 - accuracy: 0.7697 - val_loss: 0.2152 - val_accuracy: 0.9100
Epoch 2/100
11/11 [==============================] - 27s 2s/step - loss: 0.2235 - accuracy: 0.9027 - val_loss: 0.2108 - val_accuracy: 0.9400
Epoch 3/100
11/11 [==============================] - 26s 2s/step - loss: 0.1480 - accuracy: 0.9528 - val_loss: 0.1943 - val_accuracy: 0.9300
Epoch 4/100
11/11 [==============================] - 26s 2s/step - loss: 0.1153 - accuracy: 0.9657 - val_loss: 0.0916 - val_accuracy: 0.9800
Epoch 5/100
11/11 [==============================] - 26s 2s/step - loss: 0.1343 - accuracy: 0.9599 - val_loss: 0.1888 - val_accuracy: 0.9300
Epoch 6/100
11/11 [==============================] - 27s 2s/step - loss: 0.1090 - accuracy: 0.9614 - val_loss: 0.1378 - val_accuracy: 0.9700
Epoch 7/100
11/11 [==============================] - 27s 2s/step - loss: 0.1004 - accuracy: 0.9657 - val_loss: 0.1136 - val_accuracy: 0.9500
Epoch 8/100
11/11 [==============================] - 27s 2s/step - loss: 0.0890 - accuracy: 0.9700 - val_loss: 0.2006 - val_accuracy: 0.9400
Epoch 9/100
11/11 [==============================] - 26s 2s/step - loss: 0.0772 - accuracy: 0.9757 - val_loss: 0.1294 - val_accuracy: 0.9500
Epoch 10/100
11/11 [==============================] - 26s 2s/step - loss: 0.0904 - accuracy: 0.9714 - val_loss: 0.1557 - val_accuracy: 0.9500
Epoch 11/100
11/11 [==============================] - 26s 2s/step - loss: 0.0902 - accuracy: 0.9700 - val_loss: 0.1188 - val_accuracy: 0.9300
Epoch 12/100
11/11 [==============================] - 27s 2s/step - loss: 0.0767 - accuracy: 0.9771 - val_loss: 0.0865 - val_accuracy: 0.9600
Epoch 13/100
11/11 [==============================] - 27s 2s/step - loss: 0.0913 - accuracy: 0.9671 - val_loss: 0.1510 - val_accuracy: 0.9500
Epoch 14/100
11/11 [==============================] - 27s 2s/step - loss: 0.0796 - accuracy: 0.9714 - val_loss: 0.1849 - val_accuracy: 0.9300
Epoch 15/100
11/11 [==============================] - 26s 2s/step - loss: 0.0736 - accuracy: 0.9742 - val_loss: 0.1171 - val_accuracy: 0.9800
Epoch 16/100
11/11 [==============================] - 27s 2s/step - loss: 0.0667 - accuracy: 0.9757 - val_loss: 0.1399 - val_accuracy: 0.9600
Epoch 17/100
11/11 [==============================] - 26s 2s/step - loss: 0.0685 - accuracy: 0.9757 - val_loss: 0.1526 - val_accuracy: 0.9400
Epoch 18/100
11/11 [==============================] - 26s 2s/step - loss: 0.0509 - accuracy: 0.9814 - val_loss: 0.1003 - val_accuracy: 0.9800
Epoch 19/100
11/11 [==============================] - 27s 2s/step - loss: 0.0638 - accuracy: 0.9785 - val_loss: 0.1241 - val_accuracy: 0.9500
Epoch 20/100
11/11 [==============================] - 26s 2s/step - loss: 0.0465 - accuracy: 0.9914 - val_loss: 0.0839 - val_accuracy: 0.9600
```

Figure 3. Training Model Xception

```
Epoch 1/100
11/11 [==============================] - 117s 11s/step - loss: 0.7290 - accuracy: 0.5866 - val_loss: 0.5845 - val_accuracy: 0.7800
Epoch 2/100
11/11 [==============================] - 30s 3s/step - loss: 0.5561 - accuracy: 0.7310 - val_loss: 0.4734 - val_accuracy: 0.8900
Epoch 3/100
11/11 [==============================] - 30s 3s/step - loss: 0.4196 - accuracy: 0.8369 - val_loss: 0.3416 - val_accuracy: 0.8700
Epoch 4/100
11/11 [==============================] - 31s 3s/step - loss: 0.3085 - accuracy: 0.8770 - val_loss: 0.2384 - val_accuracy: 0.9400
Epoch 5/100
11/11 [==============================] - 30s 3s/step - loss: 0.2526 - accuracy: 0.9099 - val_loss: 0.3087 - val_accuracy: 0.8600
Epoch 6/100
11/11 [==============================] - 31s 3s/step - loss: 0.2286 - accuracy: 0.9084 - val_loss: 0.1878 - val_accuracy: 0.9300
Epoch 7/100
11/11 [==============================] - 30s 3s/step - loss: 0.1752 - accuracy: 0.9342 - val_loss: 0.1754 - val_accuracy: 0.9300
Epoch 8/100
11/11 [==============================] - 31s 3s/step - loss: 0.1609 - accuracy: 0.9356 - val_loss: 0.1267 - val_accuracy: 0.9600
Epoch 9/100
11/11 [==============================] - 30s 3s/step - loss: 0.1712 - accuracy: 0.9227 - val_loss: 0.1613 - val_accuracy: 0.9400
Epoch 10/100
11/11 [==============================] - 30s 3s/step - loss: 0.1367 - accuracy: 0.9471 - val_loss: 0.1835 - val_accuracy: 0.9200
Epoch 11/100
11/11 [==============================] - 30s 3s/step - loss: 0.1536 - accuracy: 0.9328 - val_loss: 0.1396 - val_accuracy: 0.9500
Epoch 12/100
11/11 [==============================] - 30s 3s/step - loss: 0.1832 - accuracy: 0.9242 - val_loss: 0.2283 - val_accuracy: 0.8900
Epoch 13/100
11/11 [==============================] - 30s 3s/step - loss: 0.1916 - accuracy: 0.9256 - val_loss: 0.2125 - val_accuracy: 0.9100
Epoch 14/100
11/11 [==============================] - 31s 3s/step - loss: 0.1539 - accuracy: 0.9413 - val_loss: 0.1541 - val_accuracy: 0.9400
Epoch 15/100
11/11 [==============================] - 30s 3s/step - loss: 0.1359 - accuracy: 0.9442 - val_loss: 0.1569 - val_accuracy: 0.9700
Epoch 16/100
11/11 [==============================] - 30s 3s/step - loss: 0.1300 - accuracy: 0.9442 - val_loss: 0.2477 - val_accuracy: 0.9000
Epoch 17/100
11/11 [==============================] - 30s 3s/step - loss: 0.1954 - accuracy: 0.9199 - val_loss: 0.3240 - val_accuracy: 0.8300
Epoch 18/100
11/11 [==============================] - 30s 3s/step - loss: 0.2056 - accuracy: 0.9142 - val_loss: 0.2462 - val_accuracy: 0.8800
Epoch 19/100
11/11 [==============================] - 30s 3s/step - loss: 0.1471 - accuracy: 0.9485 - val_loss: 0.1421 - val_accuracy: 0.9700
Epoch 20/100
11/11 [==============================] - 30s 3s/step - loss: 0.1169 - accuracy: 0.9557 - val_loss: 0.1705 - val_accuracy: 0.9400
```

Figure 4. Training Model VGG19

Figures 3 and 4 illustrate the training process of CNN models, specifically the Xception and VGG19 architectures. In this study, variations of several parameters such as *learning rate*, *dropout,* and number of *filters* are applied to explore their impact on the model performance. By conducting experiments using various combinations of parameters, researchers can understand how the model responds to such changes and determine the optimal configuration to improve accuracy. In each experiment, the Xception and VGG19 models were trained using different parameter configurations. Once the training and validation process was complete, the accuracy was measured and recorded for each trial. Subsequently, the results were evaluated and compared to identify the parameter combination that gave the best performance. In this analysis of the training results, we included graphs showing the *validation loss* and *validation accuracy* during the training process, as well as a comparison of the model performance on each parameter configuration tested. This analysis is essential for understanding how parameter variations affect model performance, detecting possible *overfitting* or *underfitting*, and finding parameter values that produce validation data with high and stable accuracy.

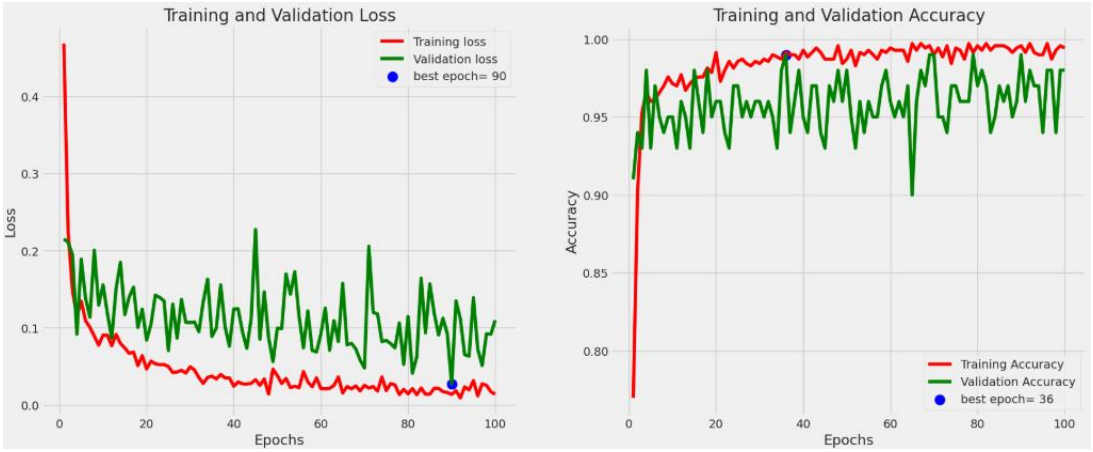

Figure 5. Xception Model Evaluation Results

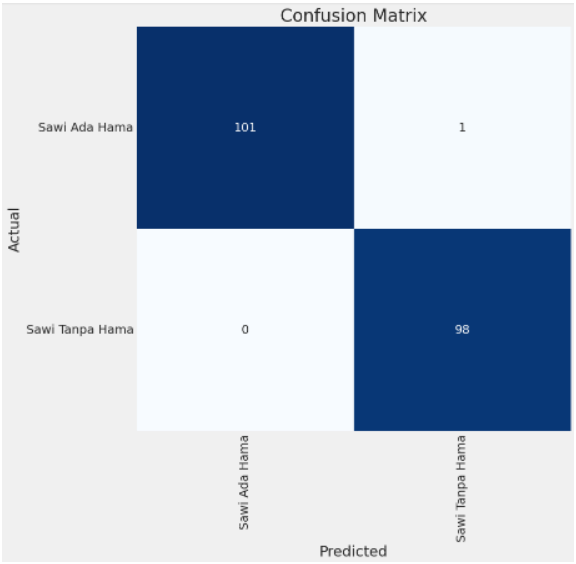

Figure 6. Confussion Matrix Model Xception

Figure 7. VGG19 Model Evaluation Results

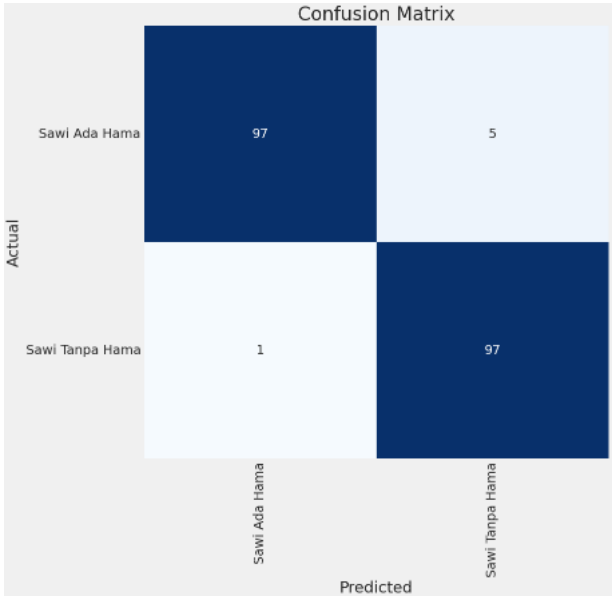

Figure 8. Confussion Matrix Model VGG19

4. CONCLUSION

Based on the experimental results described above, the Xception model shows superior performance in mustard leaf disease classification with more stable validation accuracy and lower loss than VGG19. Although both models achieved high accuracy of around 99% for Xception and 94.50% for VGG19, the VGG19 model experienced larger fluctuations in validation loss, indicating a higher potential for overfitting. In addition, in terms of time efficiency, VGG19 reaches optimal accuracy faster at the 29th epoch, while Xception requires more epochs to reach stability. Thus, Xception is a better choice in terms of accuracy and stability, while VGG19 is superior in training time efficiency. In an effort to improve the accuracy of the model in the classification of mustard leaf diseases, we recommend expanding the number of datasets or conducting a more careful selection of datasets, especially in disease classes that have similar characteristics. By improving the quality and variety of training data, the model can more effectively recognize differences between disease types, thereby reducing the risk of misclassification. In addition, the results of this study are expected to benefit the world of knowledge and also for farmers in detecting mustard leaf diseases earlier and more accurately. Faster and more precise detection enables the implementation of more optimal preventive measures, which in turn can reduce potential losses due to disease attacks and increase crop yields and overall agricultural productivity.

ACKNOWLEDGEMENTS

Author would like to express gratitude to Prof. Dr. Kusrini, M.Kom. for providing guidance throughout this research by offering valuable suggestions, insights, and knowledge, as well as for motivating the author during the completion of this study.

REFERENCES

[1] Y. T. Samiha, "Strategi Pemanfaatan Media Air (Hidroponik) Pada Budidaya Tanaman Kangkung, Pakcoy dan Sawi Sebagai Alternatif Urban Farming," *J. Educ.*, vol. 06, no. 01, pp. 5835–5848, 2023.

[2] A. Sood, P. Kumar Sarangi, A. Kumar Sahoo, L. Rani, K. Bajaj, and A. Kumar Agrawal, "AI-Driven Mustard Disease Identification: A Multiclass and Binary Classification Approach for Advanced Crop Health Monitoring," *Proc. - Int. Conf. Technol. Adv. Comput. Sci. ICTACS 2023*, pp. 256–261, 2023, doi: 10.1109/ICTACS59847.2023.10390082.

[3] M. Utami and Erwin Dwika Putra, "Deteksi Objek Kualitas Daun Sawi Menggunakan Metode HSV Color dan Color Blob," *JUSIBI (Jurnal Sist. Inf. dan Bisnis)*, vol. 5, no. 2, pp. 85–93, 2023, doi: 10.54650/jusibi.v5i2.518.

[4] Sakshi, C. Sharma, S. Sharma, T. Sharma, and S. Gochhait, "Green Intelligence: A Sequential CNN Odyssey in Mustard Leaf Disease Detection," *2024 ASU Int. Conf. Emerg. Technol. Sustain. Intell. Syst. ICETSIS 2024*, no. March, pp. 877–882, 2024, doi: 10.1109/ICETSIS61505.2024.10459509.

[5] Y. M. Abd Algani, O. J. Marquez Caro, L. M. Robladillo Bravo, C. Kaur, M. S. Al Ansari, and B. Kiran Bala, "Leaf disease identification and classification using optimized deep learning," *Meas. Sensors*, vol. 25, no. September 2022, p. 100643, 2023, doi: 10.1016/j.measen.2022.100643.

[6] M. T. A. Syech Ahmad and B. Sugiarto, "Implementasi Convolutional Neural Network (CNN) untuk Klasifikasi Ikan Cupang Berbasis Mobile," *Digit. Transform. Technol.*, vol. 3, no. 2, pp. 712–723, 2023, doi: 10.47709/digitech.v3i2.3245.

[7] C. L. Nazalia, P. Palupiningsih, B. Prayitno, and Y. S. Purwanto, "Implementation of Convolutional Neural Network Algorithm to Pest Detection in Caisim," *ICCoSITE 2023 - Int. Conf. Comput. Sci. Inf. Technol. Eng. Digit. Transform. Strateg. Facing VUCA TUNA Era*, pp. 609–614, 2023, doi: 10.1109/ICCoSITE57641.2023.10127792.

[8] C. L. Srinidhi, O. Ciga, and A. L. Martel, "Deep neural network models for computational histopathology: A survey," *Med. Image Anal.*, vol. 67, p. 101813, 2021, doi: 10.1016/j.media.2020.101813.

[9] L. Trihardianingsih, A. Sunyoto, and T. Hidayat, "Classification of Tea Leaf Diseases Based on ResNet-50 and Inception V3," *Sinkron*, vol. 8, no. 3, pp. 1564–1573, 2023, doi: 10.33395/sinkron.v8i3.12604.

[10] A. Mehrish, N. Majumder, R. Bharadwaj, R. Mihalcea, and S. Poria, *A review of deep learning techniques for speech processing*, vol. 99. 2023. doi: 10.1016/j.inffus.2023.101869.

[11] W. Bian, "Real-Fake Face Detection Based on Joint Multi-Layer CNN Structure and Data Augmentation," no. Daml 2023, pp. 25–29, 2024, doi: 10.5220/0012800300003885.

[12] J. A. AYENI, "Convolutional Neural Network (CNN): The architecture and applications," *Appl. J. Phys. Sci.*, vol. 4, no. 4, pp. 42–50, 2022, doi: 10.31248/ajps2022.085.

[13] A. Mumuni and F. Mumuni, "Data augmentation: A comprehensive survey of modern approaches," *Array*, vol. 16, no. November, p. 100258, 2022, doi: 10.1016/j.array.2022.100258.

[14] S. M. Hassan and A. K. Maji, "Plant Disease Identification Using a Novel Convolutional Neural Network," *IEEE Access*, vol. 10, pp. 5390–5401, 2022, doi: 10.1109/ACCESS.2022.3141371.

[15] J. Kotwal, D. R. Kashyap, and D. S. Pathan, "Agricultural plant diseases identification: From traditional approach to deep learning," *Mater. Today Proc.*, vol. 80, no. March, pp. 344–356, 2023, doi: 10.1016/j.matpr.2023.02.370.