IJCCS (Indonesian Journal of Computing and Cybernetics Systems)
Vol.17, No.1, January 2023, pp. 103~114
ISSN (print): 1978-1520, ISSN (online): 2460-7258
DOI: https://doi.org/10.22146/ijccs.80956

Comparison of CNN Models With Transfer Learning in the Classification of Insect Pests

Angga Prima Syahputra[1], Alda Cendekia Siregar*[2], Rachmat Wahid Saleh Insani[3]
[1]Department of Computer Science and Electronics, Pontianak Muhammadiyah University, Pontianak, Indonesia
e-mail: [1]181220035@unmuhpnk.ac.id, ***[2]Alda.siregar@unmuhpnk.ac.id** ,
[3]Rachmat.wahid@unmuhpnk.ac.id

Abstract

Hama serangga merupakan masalah yang penting untuk diatasi dalam bidang pertanian. Tujuan dari penelitian ini adalah mengklasifikasi hama serangga dengan dataset ip-102 dengan menggunakan beberapa pre-trained model CNN dan memilih model mana yang terbaik dalam mengklasifikasi data hama serangga. Metode yang digunakan yaitu metode transfer learning dengan pendekatan fine-tuning. Transfer learning dipilih karena teknik ini dapat menggunakan fitur dan bobot yang telah diperoleh selama proses pelatihan sebelumnya. Dengan demikian, waktu komputasi dapat dikurangi dan akurasi dapat meningkat. Model yang digunakan diantaranya Xception, MobileNetV3L, MobileNetV2, DenseNet-201, dan InceptionV3. Teknik fine-tuning dan freeze layer juga digunakan untuk meningkatkan kualitas model yang dihasilkan, menjadikannya lebih akurat dan lebih sesuai dengan masalah yang dihadapi. Penelitian ini menggunakan 75.222 data citra dari 102 kelas. Hasil dari penelitian ini adalah model DenseNet-201 dengan fine-tuning menghasilkan nilai akurasi 70%. MobileNetV2 dengan fine-tuning menghasilkan nilai akurasi 66%. MobileNetV3L dengan fine-tuning menghasilkan nilai akurasi 68%. InceptionV3 dengan fine-tuning menghasilkan nilai akurasi 67%. Xception dengan fune-tuning menghasilkan nilai akurasi 69%. Kesimpulan dari penelitian ini adalah metode transfer learning dengan pendekatan fine-tuning menghasilkan nilai akurasi tertinggi yaitu sebesar 70% pada model DenseNet-201.

Keywords*— CNN, Transfer learning, fine-tuning, Klasifikasi, hama serangga*

Abstract

Insect pests are an important problem to overcome in agriculture. The purpose of this research is to classify insect pests with the IP-102 dataset using several CNN pre-trained models and choose which model is best for classifying insect pest data. The method used is the transfer learning method with a fine-tuning approach. Transfer learning was chosen because this technique can use the features and weights that have been obtained during the previous training process. Thus, computation time can be reduced and accuracy can be increased. The models used include Xception, MobileNetV3L, MobileNetV2, DenseNet-201, and InceptionV3. Fine-tuning and freeze layer techniques are also used to improve the quality of the resulting model, making it more accurate and better suited to the problem at hand. This study uses 75,222 image data with 102 classes. The results of this study are the DenseNet-201 model with fine-tuning produces an accuracy value of 70%, MobileNetV2 66%, MobileNetV3L 68%, InceptionV3 67%, Xception 69%. The conclusion of this study is that the transfer learning method with the fine-tuning approach produces the highest accuracy value of 70% in the DenseNet-201 model.

Keywords*— CNN, transfer learning, fine-tuning, classification, Insect pest*

Received January 4th,2023; Revised January 31th, 2023; Accepted January 31th, 2023

1. INTRODUCTION

Insect pests can significantly damage crops and harm agricultural production. These pests can cause significant losses in crops like rice, wheat, and other economically valuable crops, leading to crop failures [1]. Farmers often face numerous challenges in trying to achieve good crop yields. One of these challenges is pest control, as pests can cause significant losses to farmers.

Preventing crop damage from insect pests starts with the ability to identify these pests. To do so, farmers must be able to differentiate between insects that cause harm to their crops and those that do not. One solution to identify insect pests is through the use of deep learning methods, particularly Convolutional Neural Networks (CNN). These networks are currently the most effective choice for object detection tasks such as insect image recognition and classification. However, sometimes the available data may be limited, making it difficult to train a CNN from scratch. In these cases, transfer learning techniques can be used to take advantage of the capabilities of cnns while conserving computational resources [2].

In a previous study by Xiaoping Wu, et al [1], the performance of the IP-102 dataset was tested using modern machine learning methods, including four deep learning models: Alexnet, Googlenet, Vggnet, and Resnet. The Resnet model achieved the highest accuracy value at 49%.

In a previous study by Loris Nanni, et al [3], the classification of insect pests was explored using saliency methods and convolutional neural networks. This study tested two small datasets and the IP-102 dataset. The highest accuracy value achieved using the CNN method on the IP-102 dataset was 61.93%.

Previous research by Khalifa, et al [4], classified insect pests using the transfer learning method and data augmentation techniques. The IP-102 dataset, containing 27,500 images, was used for this study. The models tested included Alexnet, Googlenet, and Squeeznet. The selection of the model architecture was based on the number of layers in the architecture. The performance of the selected model was evaluated using test accuracy and performance metrics such as precision, recall, and F1 score. The Alexnet model achieved the highest test accuracy at 89.33%. Additionally, the Alexnet model had the smallest number of layers, which reduced training time and computational complexity.

The purpose of this research is to classify insect pests using the IP-102 dataset which consists of 75,222 images. Transfer learning is utilized in this study, a technique that involves using and adjusting pre-trained networks. In this method, a previously trained model serves as the foundation for adding feature extraction layers to the beginning and middle layers, as well as replacing the final layer for the classification of insect pests [5]. The pre-trained models used include Xception, MobileNetV3L, MobileNetV2, DenseNet-201, and InceptionV3.

However, while the use of transfer learning can increase the accuracy of the pre-trained model for new tasks, it is necessary to unfreeze some of the layers in the model to be retrained while keeping the layers above unfrozen. This process is typically done through a fine-tuning approach [6]. Fine-tuning is used to improve the classification performance of the chosen model. By achieving good results, this research can determine which model has the best performance in classifying insect pests.

2. METHODS

The transfer learning method is used to classify insect pest image data with the IP-102 dataset. Figure 1 is a flowchart of research on the classification of insect pests.

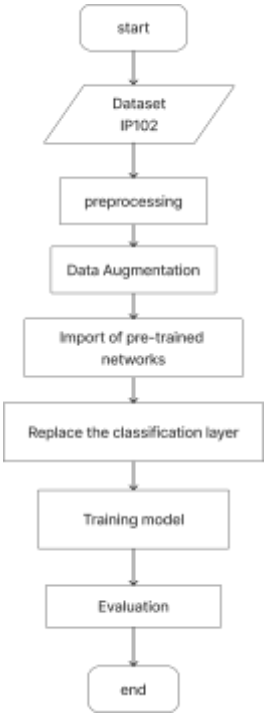

Figure 1 Flow chart of research on the classification of insect pests

2. 1 Datasets

To get started, the first way is to collect a dataset. The dataset used in this study is a public dataset that can be accessed via the Kaggle website under the name IP-102-DATASET. This dataset was created by Wu, et al [1]. The dataset contains images of insect pests consisting of validation images, training images and testing images. IP-102 contains 75,222 image data used in this study. Table 1 shows the amount of each IP-102 data.

Table 1 Number of IP-102 Datasets

Data sets	Amount of data
Training data	45095
Data validation	7508
Test data	22619
Total	75,222

2. 2 Preprocessing

Preprocessing is done on insect pest image data. The dataset is divided into three parts, namely 70% training data, 10% validation data, and 30% testing data. The image has three color channels, namely red, green, and blue (RGB). Image sizes vary, so the resolution is changed to 224 × 224. In addition, the image is also normalized by dividing the intensity value of each pixel by 255 so that the range of values for each pixel is -1 to 1 [7]. Mathematically normalization is calculated by equation (1)

$$Normalization = \frac{X - \mu}{\Sigma} \quad (1)$$

Where x is the original feature vector, μ is the mean, and σ is the standard deviation[8].
Figure 2 shows the preprocessing process flow

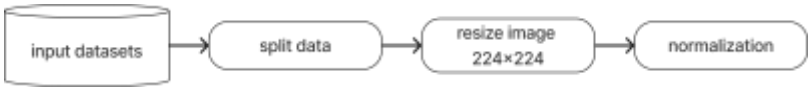

Figure 2 Pre-processing flow chart

2. 3 Data augmentation

To make the model built better and enable it to benefit from the little data set by expanding it, data augmentation is used to increase the size and quality of the training data [9]. Data augmentation is done by applying horizontal and vertical random flips, as well as random rotation with a value of 0.2. This is done so that the model can study images with different transformations. That way, the model will be better able to recognize the image of insect pests with various variations.

2. 4 Import Pre-Trained Networks

This study uses the Xception, MobileNetV3L, MobileNetV2, DenseNet-201, and InceptionV3 models as pre-trained models. The pre-trained models were trained using the Imagenet dataset, which includes features ranging from simple ones such as brightness and borders, to more complex and unique features such as color and shape [5]. Transfer learning can be used to classify insect pests in the target domain by applying the results from the feature extraction layer in the source domain to the feature extraction layer in the target domain. This is illustrated in the framework shown in Figure 3.

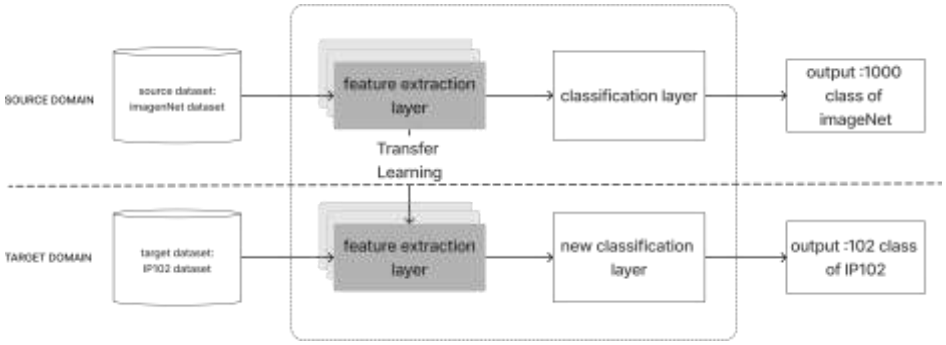

Figure 3 Frameworks from transfer learning for classifying insect pests

2.4.1 Xception

This model consists of several layers called "Separable Convolution blocks", each of which consists of several convolutional layers separated into two parts: Depthwise convolution, and Pointwise convolution. This allows the model to capture information on a smaller scale with a lower number of parameters, thereby increasing computational efficiency and accuracy. Xception also uses a data normalization technique called "batch normalization" to reduce accuracy depending on the input scale. This model has proven effective in a variety of machine learning tasks, including image classification and object recognition [10].

2.4.2 MobileNetV2

This model is the latest version of MobileNet which is intended for use on mobile devices with limited computing power. MobileNetV2 uses a technique called "inverted residuals" which allows the model to combine information from the layers connected to it and improve accuracy. This model also uses a technique called "linear bottlenecks" which improves computational efficiency by reducing the number of parameters that must be optimized. MobileNetV2 has proven effective in a variety of machine learning tasks, including image classification and object recognition [11].

2.4.3 MobileNetV3L

This model is the latest version of MobileNet using an architecture called "MobileNetV3", which is intended for use in mobile devices with limited computing power. MobileNetV3-Large uses a technique called "squeeze-and-excitation" which allows the model to combine information from the layers connected to it and improve accuracy. The model also uses a technique called "residual connections" which allows the model to capture information lost during the convolution process and improve computational efficiency. MobileNetV3-Large has proven effective in a variety of machine learning tasks, including image classification and object recognition [12].

2.4.4 DenseNet-201

This model consists of layers called "Dense blocks", each of which consists of multiple convolutional layers directly connected to all previous layers in the block. This allows all layers to receive information from across the network, reducing the need to discipline unrelated attributes and improving the transfer of information across the network. DenseNet-201 has 201 layers and has proven effective in a variety of machine learning tasks, including image classification and object recognition [13].

2.4.5 Inception V3

This model consists of several layers called "Inception blocks", each of which consists of several convolutional layers with different sized filters that run in parallel, then sum together. This allows the model to capture information at multiple scales simultaneously, thereby improving computational efficiency and increasing accuracy. Inception V3 also uses a data normalization technique called "batch normalization" to reduce accuracy depending on the scale of the input. This model has proven effective in a wide range of machine learning tasks, including image classification and object recognition [14].

2. 5 Replace The Classification Layer

During the training of a model with the transfer learning approach, the last layer of the previously trained model must be replaced with a new classification layer. The layers added to the end of the CNN are trained first, while the weights in the feature extraction section are kept unchanged. This allows the model to focus on training the new classification layer, enabling it to better learn the task at hand [15]. All pre-trained models used will have a Global Average Pooling layer, Dropout with a value of 0.3, and an output layer with 102 classes and a Softmax activation function added to them [4]. The Softmax activation function is used in the output layer because it is the only activation function recommended for multi-label classification [13]. Figure 4 shows the transfer learning process with a pre-trained model and the addition of a new layer. The image is an additional classification layer

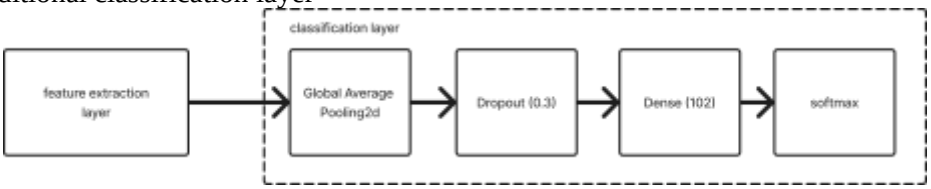

Figure 4 Added classification layer

2. 6 Training Models

At the training stage, two methods are used, namely without using fine-tuning and using fine-tuning [16]. Fine-tuning functions to train the pre-trained model by training several layers of the base model to achieve better accuracy [17]. In the model training process, the training process is carried out 2 times by training without unfreezing the model to see the model's performance. If

the process does not get satisfactory results, then a retraining process is carried out by opening several layers in the pre-trained model to be retrained.

2. 7 Evaluation

To measure the performance of the model used for classification, the Confusion matrix (CM) is used. Model evaluation is done by monitoring the number of true positives, true negatives, false positives, and false negatives [18]. From these data, accuracy, precision, recall, and F1 score can be calculated [4]. Each is presented from equation (2) to equation (5).

$$Accuracy = \frac{(TN+TP)}{(TN+TP+FN+FP)} \quad (2)$$

$$Precision = \frac{TP}{(TP+FP)} \quad (3)$$

$$\text{Recall} = \frac{TP}{(TP+FN)} \quad (4)$$

$$\text{F1 Score} = 2 \times \frac{Precision \times Recall}{(Precision + Recall)} \quad (5)$$

The following terms are used to describe the performance of the model:
a. True positive (TP): positive insect pest image data that is correctly predicted
b. True negative (TN): negative insect pest image data that is correctly predicted
c. False positive (FP): negative insect pest im age data that is incorrectly predicted as positive
d. False negative (FN): positive insect pest image data that is incorrectly predicted as negative

3. RESULTS AND DISCUSSION

3.1 Training results

The training process is conducted in two stages. The Adam Optimizer is used as the optimizer and the Loss function is the Category Cross Entropy. The first stage consists of 50 epochs of training without fine-tuning, using a learning rate of 0.0001. The second stage includes 50 epochs of training with fine-tuning, with a learning rate of 0.00001, which is one-tenth of the learning rate used in the first stage. Figure 5 to Figure 9 show the results of training with five different architectural models using the IP-102 dataset.

3.1.1 Xception

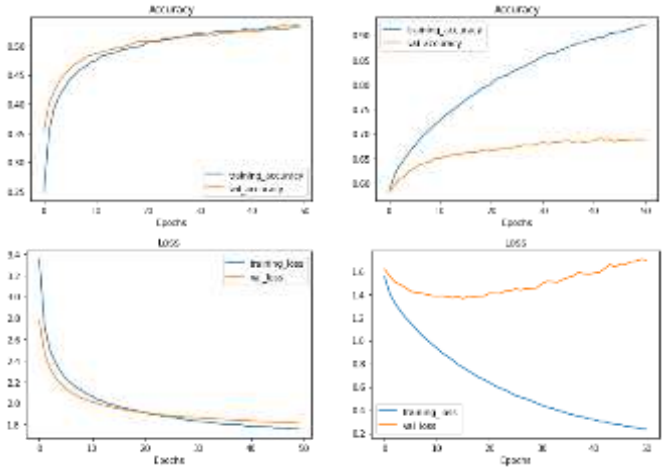

Figure 5 A comparison of the model's accuracy and loss for each epoch using the pre-trained Xception layer, with and without fine-tuning, is shown (left: without fine-tuning, right: with fine-tuning)

3.1.2 MobileNetV3L

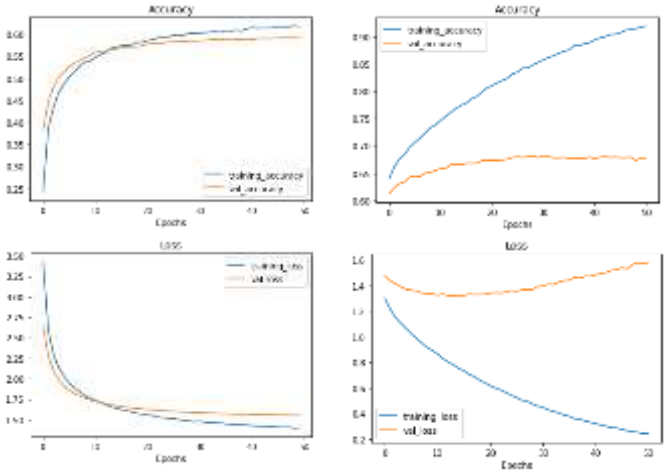

Figure 6 A comparison of the model's accuracy and loss for each epoch using the pre-trained MobileNetV3L layer, with and without fine-tuning, is shown (left: without fine-tuning, right: with fine-tuning)

3.1.3 MobileNetV2

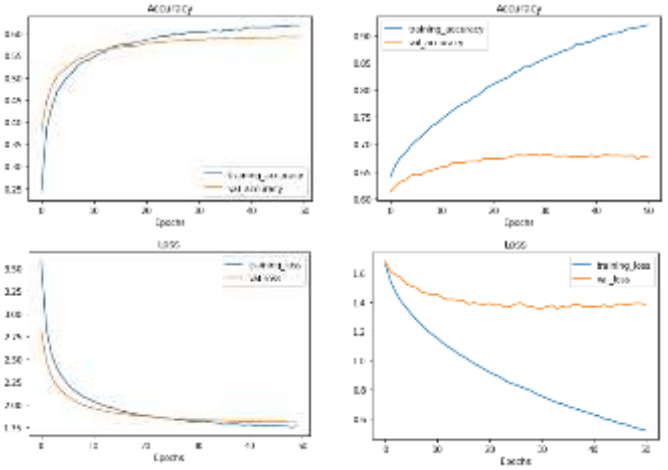

Figure 7 A comparison of the model's accuracy and loss for each epoch using the pre-trained MobileNetV2 layer, with and without fine-tuning, is shown (left: without fine-tuning, right: with fine-tuning)

3.1.4 DenseNet-201

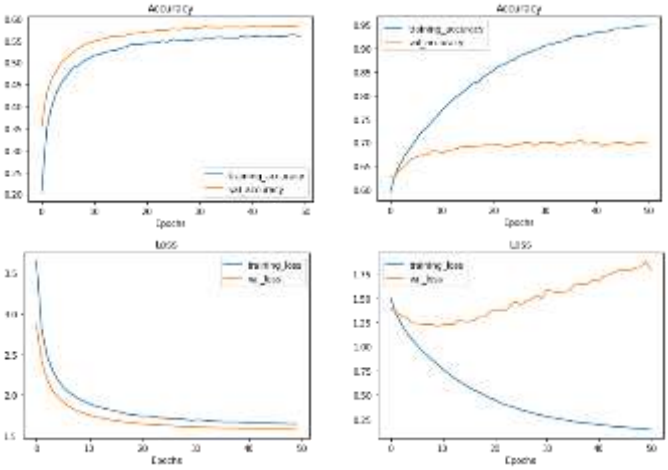

Figure 8 A comparison of the model's accuracy and loss for each epoch using the pre-trained DenseNet-201 layer, with and without fine-tuning, is shown (left: without fine-tuning, right: with fine-tuning)

3.1.5 *InceptionV3*

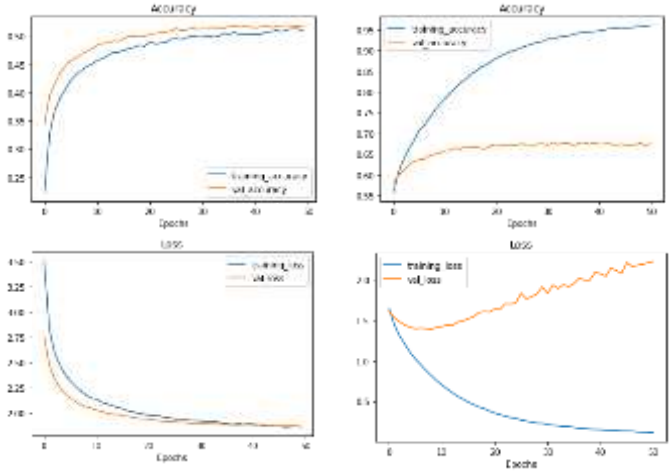

Figure 9 A comparison of the model's accuracy and loss for each epoch using the pre-trained InceptionV3 layer, with and without fine-tuning, is shown (left: without fine-tuning, right: with fine-tuning)

Accuracy training. Each model has a different number of parameters. The variance in the total parameters of each architectural trained model via fine-tuning is shown in Table 2

Table 2 Total training parameter fine-tuning

Model	Total params	Trainable params
MobileNetV2	2,388,646	2,305,254
MobileNetV3L	4,357,094	4,308,758
DesNet201	18,517,926	17,943,782
Xception	21,070,478	17,757,222
InceptionV3	22,011,782	21,465,462

The MobileNetV2 and MobileNetV3L models have the least number of parameters, while the InceptionV3, Xception, and DesNet201 models have a larger number of parameters. The accuracy difference between models trained with and without fine-tuning is shown in Figure 10.

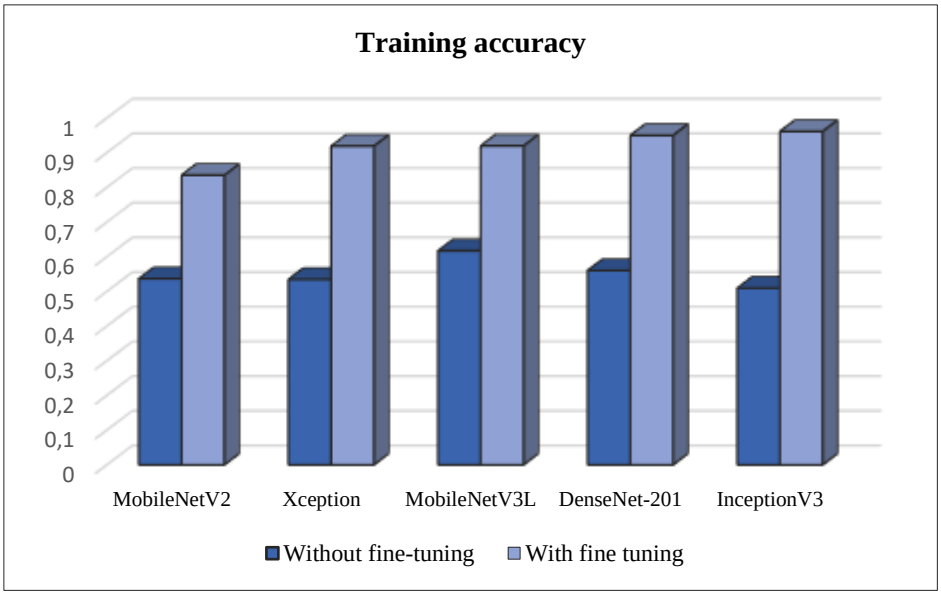

Figure 10 Differences in accuracy training results

3.2 Test result

Test results are obtained using a confusion matrix on the testing data. The testing is performed with and without fine-tuning. Tables 3 and 4 present the performance metrics for various CNN models.

Table 3 Performance metrics without fine-tuning

Model	Accuracy	Precision	Recall	F1-score
InceptionV3	0.52	0.51	0.52	0.50
Xception	0.54	0.52	0.54	0.51
MobileNetV2	0.54	0.53	0.54	0.52
DenseNet-201	0.59	0.58	0.59	0.57
MobileNetV3L	0.59	0.58	0.59	0.58

Table 4 Performance metrics with fine-tuning

Model	*Accuracy*	Precision	Recall	F1-score
MobileNetV2	0.66	0.66	0.66	0.65
InceptionV3	0.67	0.67	0.67	0.67
MobileNetV3L	0.68	0.68	0.68	0.68
Xception	0.69	0.68	0.69	0.68
DenseNet-201	0.70	0.70	0.70	0.70

According to Table 3, MobileNetV3L has the highest percentage for accuracy, precision, recall, and f1-score when not using fine-tuning. MobileNetV3L's f1-score is 1% higher than that of DenseNet-201. Table 4 demonstrates that DenseNet-201 achieves the highest percentage for accuracy, precision, recall, and f1-score, with all results at 70%, when using the fine-tuning approach. The difference in accuracy between models tested with and without fine-tuning is shown in Figure 11.

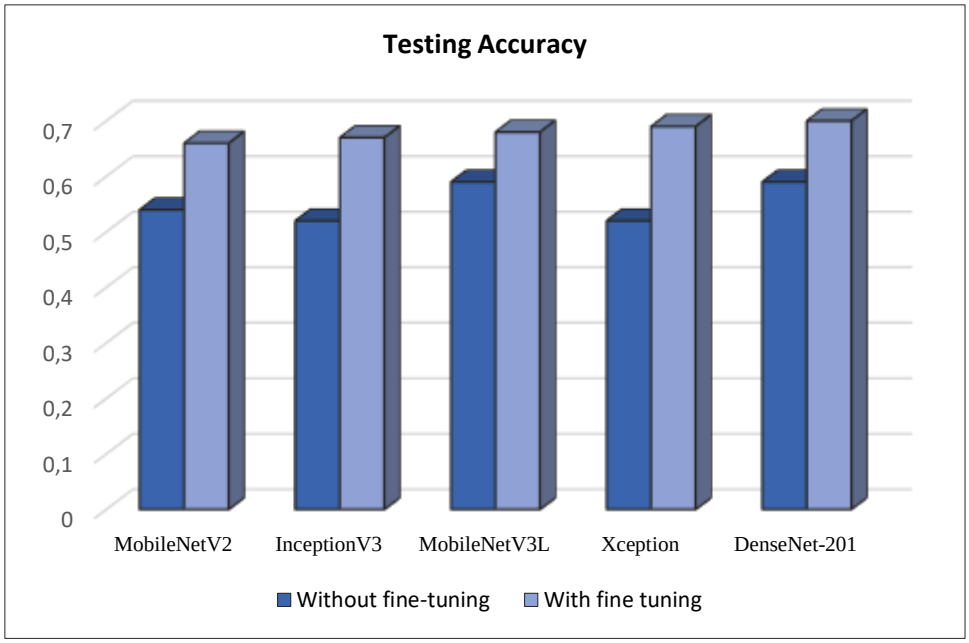

Figure 11 Differences in accuracy testing results

After knowing that DenseNet-201 gives the best results co mpared to other models, then try to make predictions with 18 insect pest image data to see the results of the predictions with the original label. From 18 image data, 11 image data are predicted to be correct. Figure 12 shows the prediction results with test data using the DenseNet-201 model

Figure 12 Example of testing the accuracy of classification using DenseNet-201

4. CONCLUSIONS

The research shows that using the transfer learning method with five CNN architectural models to classify insect pest images with the IP-102 dataset produces the highest accuracy when using the DenseNet-201 model, with an accuracy of 70% using fine-tuning and 59% without fine-tuning. Additionally, the MobileNetV2 model has an accuracy of 66% with fine-tuning and 54% without fine-tuning. The MobileNetV3L model has an accuracy of 68% with fine-tuning and 59% without fine-tuning. The InceptionV3 model has an accuracy of 67% with fine-tuning and 52% without fine-tuning. The Xception model has an accuracy of 69% with fine-tuning and 54% without fine-tuning.

REFERENCES

[1] X. Wu, C. Zhan, Y. K. Lai, M. M. Cheng, and J. Yang, "IP102: A large-scale benchmark dataset for insect pest recognition," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2019-June, pp. 8779–8788, 2019, doi: 10.1109/CVPR.2019.00899.

[2] D. M. Wonohadidjojo, "Perbandingan Convolutional Neural Network pada Transfer Learning Method untuk Mengklasifikasikan Sel Darah Putih," *Ultim. J. Tek. Inform.*, vol. 13, no. 1, pp. 51–57, 2021, doi: 10.31937/ti.v13i1.2040.

[3] L. Nanni, G. Maguolo, and F. Pancino, "Insect pest image detection and recognition based on bio-inspired methods," *Ecol. Inform.*, vol. 57, 2020, doi: 10.1016/j.ecoinf.2020.101089.

[4] N. E. M. Khalifa, M. Loey, and M. H. N. Taha, "Insect pests recognition based on deep transfer learning models," *J. Theor. Appl. Inf. Technol.*, vol. 98, no. 1, pp. 60–68, 2020.

[5] R. Setiawan, Wahyudi Utoyo, Moh Imam Rulaningtyas, "Transfer learning with multiple pre-trained network for fundus classification," *Telkomnika (Telecommunication Comput. Electron. Control.*, vol. 18, no. 3, pp. 1382–1388, 2020, doi: 10.12928/TELKOMNIKA.v18i3.14868.

[6] G. Vrbančič and V. Podgorelec, "Transfer learning with adaptive fine-tuning," *IEEE Access*, vol. 8, no. December, pp. 196197–196211, 2020, doi: 10.1109/ACCESS.2020.3034343.

[7] L. Huang, J. Qin, Y. Zhou, F. Zhu, L. Liu, and L. Shao, "Normalization Techniques in Training DNNs : Methodology , Analysis and Application," pp. 1–20.

[8] R. Kalaiarasan, K. Madhan Kumar, S. Sridhar, and M. Yuvarai, "Deep Learning-based Transfer Learning for Classification of Skin Cancer," *Proc. - Int. Conf. Appl. Artif. Intell. Comput. ICAAIC 2022*, pp. 450–454, 2022, doi: 10.1109/ICAAIC53929.2022.9792651.

[9] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *J. Big Data*, vol. 6, no. 1, 2019, doi: 10.1186/s40537-019-0197-0.

[10] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 1800–1807, 2017, doi: 10.1109/CVPR.2017.195.

[11] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 4510–4520, 2018, doi: 10.1109/CVPR.2018.00474.

[12] A. Howard *et al.*, "Searching for mobileNetV3," *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2019-Octob, pp. 1314–1324, 2019, doi: 10.1109/ICCV.2019.00140.

[13] M. F. Naufal and S. F. Kusuma, "Pendeteksi Citra Masker Wajah Menggunakan CNN dan Transfer Learning," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 8, no. 6, p. 1293, 2021, doi: 10.25126/jtiik.2021865201.

[14] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 2818–2826, 2016, doi: 10.1109/CVPR.2016.308.

[15] J. E. Widyaya and S. Budi, "Pengaruh Preprocessing Terhadap Klasifikasi Diabetic

Retinopathy dengan Pendekatan Transfer Learning Convolutional Neural Network," *J. Tek. Inform. dan Sist. Inf.*, vol. 7, no. 1, pp. 110–124, 2021, doi: 10.28932/jutisi.v7i1.3327.

[16] A. Informatics, H. Noprisson, M. Buana, and A. Info, "Fine-Tuning Model Transfer Learning VGG16 Untuk Klasifikasi Citra Penyakit Tanaman Padi," vol. 5, no. 3, pp. 244–249, 2022.

[17] E. I. Haksoro and A. Setiawan, "Pengenalan Jamur Yang Dapat Dikonsumsi Menggunakan Metode Transfer Learning Pada Convolutional Neural Network," *J. ELTIKOM*, vol. 5, no. 2, pp. 81–91, 2021, doi: 10.31961/eltikom.v5i2.428.

[18] R. Rismiyati and A. Luthfiarta, "VGG16 Transfer Learning Architecture for Salak Fruit Quality Classification," *Telematika*, vol. 18, no. 1, p. 37, 2021, doi: 10.31315/telematika.v18i1.4025.