Jurnal Teknik Informatika (JUTIF)
Vol. 5, No. 3, June 2024, pp. 685-692

DOI: https://doi.org/10.52436/1.jutif.2024.5.3.1583
p-ISSN: 2723-3863
e-ISSN: 2723-3871

OPTIMIZING BUTTERFLY CLASSIFICATION THROUGH TRANSFER LEARNING: FINE-TUNING APPROACH WITH NASNETMOBILE AND MOBILENETV2

Ni Kadek Devi Adnyaswari Putri[*1], Ardytha Luthfiarta[2], Permana Langgeng Wicaksono Ellwid Putra[3]

[1,2,3]Study Program in Informatics Engineering, Faculty of Computer Science, Universitas Dian Nuswantoro, Indonesia
Email: [1]adnyaswaridevi386@gmail.com, [2]ardytha.lutfiarta@dsn.dinus.ac.id, [3]langgeng86@gmail.com

(Article received: December 06, 2023; Revision: January 15, 2024; published: May 18, 2024)

Abstract

Butterflies play a significant role in ecosystems, especially as indicators of the state of biological balance. Each butterfly species is distinctly different, although some also show differences with very subtle traits. Etymologists recognize butterfly species through manual taxonomy and image analysis, which is time-consuming and costly. Previous research has tried to use computer vision technology, but it has shortcomings because it uses a small distribution of data, resulting in a lack of programs for recognizing various other types of butterflies. Therefore, this research is made to apply computer vision technology with the application of transfer learning, which can improve pattern recognition on image data without the need to start the training process from scratch. Transfer learning has a main method, which is fine-tuning. Fine-tuning is the process of matching parameter values that match the architecture and freezing certain layers of the architecture. The use of this fine-tuning process causes a significant increase in accuracy. The difference in accuracy results can be seen before and after using the fine-tuning process. Thus, this research focuses on using two Convolutional Neural Network architectures, namely MobileNetV2 and NASNetMobile. Both architectures have satisfactory accuracy in classifying 75 butterfly species by applying the transfer learning method. The results achieved on both architectures using fine-tuning can produce an accuracy of 86% for MobileNetV2, while NASNetMobile has a slight difference in accuracy of 85%.

Keywords: *butterfly classification, fine-tuning, MobileNetV2, NASNetMobile, transfer learning.*

1. INTRODUCTION

Butterflies are a type of insect that is spread all over the world. Butterflies have an important role in the ecosystem, which includes functions as flower pollinators, food sources, and indicators of biological well-being[1], [2]. The most striking feature of butterflies is the beauty and uniqueness of their stunning wings, which are characterized by a wide variety of colors and interesting patterns. Each type of butterfly has striking differences, in terms of body structure, wings, and antennae. There are many different species of butterflies, and most of them exhibit very subtle characteristic differences, especially in their wings[3]. Therefore, to be able to differentiate and identify specific butterfly species, only a few resources can help, due to the large number of variations and similarities between species. Etymologists identify butterflies by involving taxonomy and manual image processing, a process that is time-consuming and costly[4]. Therefore, alternative methods are needed to identify different types of butterflies. One of them is by using computer vision technology[5].

Computer vision is an interdisciplinary field of study, utilizing the ability of computers to process and identify images, and making significant contributions to the evolution of artificial intelligence technology[5]. In this context, the important role of computer vision can be seen in the process of analyzing and understanding patterns in image data[6]. As an example of its implementation, Convolutional Neural Network (CNN) is used in the classification process, where this approach uses training data to identify the most representative image features to describe a particular image class[7]. However, the use of CNNs is often faced with certain challenges, mainly related to the need for adequate hardware, such as a Graphic Processing Unit (GPU), as it involves large-scale matrix computation and optimization[8]. To overcome this obstacle, the concept of transfer learning emerged, which allows CNNs to use pre-trained models that already understand the general features of the image data. Transfer learning refers to the use of models that have been pre-trained on a specific image processing task, and then adapted for a new task, often by utilizing datasets such as ImageNet[9]. Some of the architectures developed in the context of transfer learning involve MobileNet, ResNet, NASNet, EfficientNet, VGG, and others[10]. These concepts enable high efficiency in pattern recognition on image data without the need to start the training process from scratch, accelerating model development and improving performance in various computer vision tasks[11].

Several previous researches have used transfer learning methods to classify butterfly images. One such research that covers this can be found in[12]. In that research, classification was performed on 17,769 butterfly images that were divided into 10 different classes. The classification process was performed by utilizing the VGG16, VGG19, and ResNet50 architectures. The results showed that the highest testing accuracy was achieved by VGG16, which amounted to 79.5%, followed by VGG19 with 77.2%, and ResNet50 with 70.2%. Furthermore, in another research[1], butterfly classification using the GoogleNet CNN architecture was conducted. The dataset used consisted of 120 images divided into 4 different species. The division of the dataset is done by allocating 80% for training (train) and 20% for testing (test). The results showed that the classification accuracy reached 97.5%, demonstrating the effectiveness of the GoogleNet CNN architecture in identifying and classifying butterfly species on the given dataset.

From the discussion of previous research, it can be seen that these studies have shortcomings in the distribution of datasets. The dataset distribution used in research[12] only uses a dataset distribution consisting of 10 classes of butterfly species, while in research[1] only uses 4 classes of butterfly species. The use of limited datasets can result in the program only being able to recognize a few different types of butterflies, even though there are many different types of butterflies. Therefore, this research aims to explore a wider dataset of butterfly types so that it can provide convenience for etymologists in identifying various other types of butterflies.

Based on these considerations, this research proposes a butterfly classification method by utilizing a transfer learning approach using MobileNetV2 and NASNetMobile architectures. The selection of these architectures refers to comparisons that have been made in previous studies, especially in[13]. The research evaluated the classification of honeybee comb cells using various Convolutional Neural Network (CNN) architectures such as DenseNet, Inception, MobileNet, NasNetMobile, ResNet, and Xception. The best results were obtained by the MobileNet architecture, achieving an f1-score of 94.3%. Meanwhile, another research, namely[14], also evaluated the classification of cephalopod objects using Inception, ResNet50, MobileNetV2, Xception, NASNetMobile, and DenseNet201 architectures. In the research, these architectures went through a tuning process, and the best results of the tuning were achieved by MobileNetV2 and NASNetMobile with an accuracy rate of 89.74%. Both architectures will undergo hyperparameter tuning, a step to optimize the feature representation to fit the task at hand. This process helps overcome overfitting and underfitting issues while utilizing knowledge from previous tasks. In addition, the architecture will also be subjected to layer freezing, a technique where some layers of the model remain unchanged during the fine-tuning process. This aims to ensure that the knowledge gained from previous tasks is preserved. Particularly in the early convolutional layers, freezing layers allow the model to focus more on adjustments in the later layers that are more specific to the new dataset or task at hand. The main objective of this research is to analyze the performance of both architectures in the context of the butterfly classification task before and after undergoing the tuning process.

2. METHOD

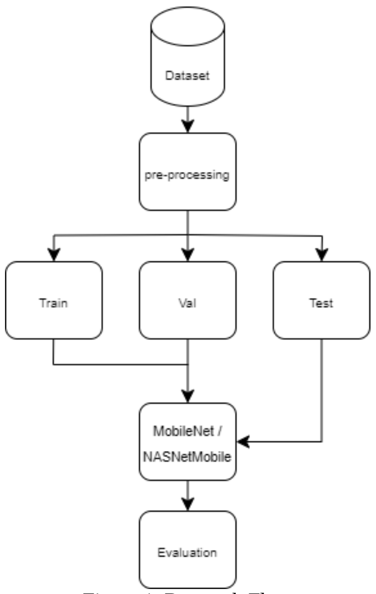

Figure 1. Research Flow

As seen in Figure 1, this research starts with collecting the dataset. After that, the dataset will undergo a preprocessing stage and be divided into three parts, namely for training, validation, and testing purposes. The training and validation datasets will be applied in the training process using the proposed architectures, namely MobileNetV2 and NASNetMobile. Meanwhile, the testing dataset will be used to evaluate the training results and to obtain the accuracy, precision, recall, and f1-score metrics of the developed model[15].

2.1. Dataset

This research utilizes a public dataset taken from the Kaggle source (https://www.kaggle.com/datasets/phucthaiv02/butterfly-image-classification)[16]. This dataset consists of 6,499 butterfly images covering 75 different species, shown in Figure 3. All images in this dataset have dimensions of 224x224 pixels. The class distribution in this dataset is unbalanced, the class with the highest number of images is Mourning Cloak, which includes 131 images, while the class with the lowest number of images is Wood Satyr, with 71 images, shown in Figure 2.

DOI: https://doi.org/10.52436/1.jutif.2024.5.3.1583
p-ISSN: 2723-3863
e-ISSN: 2723-3871

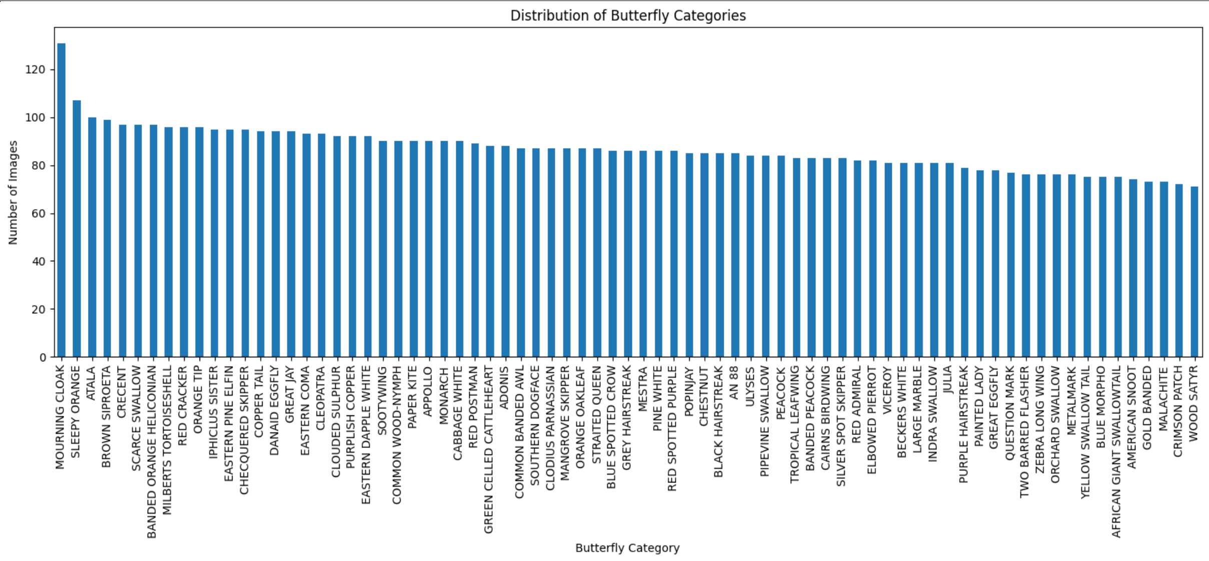

Figure 2. Dataset Distribution

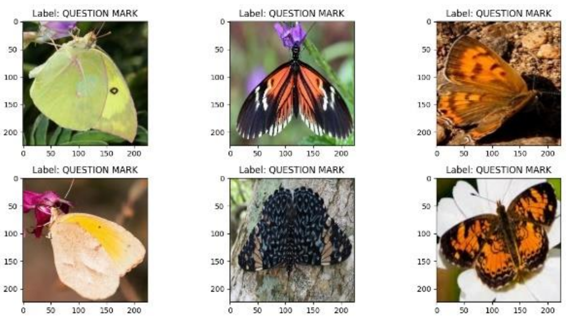

Figure 3. Dataset Sample

2.2. Preprocessing

Before proceeding to the training stage, the dataset undergoes pre-processing. This pre-processing stage involves image normalization, aiming to achieve uniform data distribution as well as ensuring consistency of variable scales within the entire dataset. Next, the dataset will be divided into three parts, with an allocation of 70% for the training dataset, 15% for the validation dataset, and 15% for the evaluation dataset, as depicted in Figure 4.

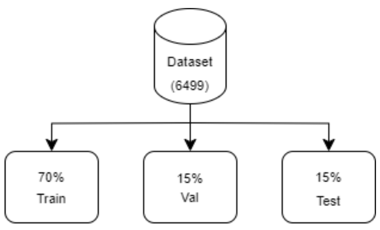

Figure 4. Dataset Splitting Flow

2.3. MobileNetV2 and NASNetMobile

The first architecture proposed in this research is MobileNetV2. The MobileNetV2 architecture is known as a lightweight architecture with fewer layers, specifically designed for mobile and embedded devices. As detailed in Table 1, the original MobileNetV2 architecture has a fully connected layer with 32 filters, followed by 19 residual bottleneck layers[8]. In addition to these layers, there is also ReLU6, a dropout layer, an average pooling layer, and batch normalization[15].

Table 1. The Original Architecture of MobileNetV2

Input	Operator	Output
2242 x 3	conv2d	32
1122 x 32	Bottleneck	16
1122 x 16	Bottleneck	24
562 x 24	Bottleneck	32
282 x 32	Bottleneck	64
142 x 64	Bottleneck	96
142 x 96	Bottleneck	160
72 x 160	Bottleneck	320
72 x 1280	Bottleneck	1280
72 x 1280	Bottleneck	-
1 x 1 x 1280	Conv2d 1 x 1	k

The next architecture proposed is NasNetMobile, a concept generated by Google's development team to find optimal parameters and create superior models. Developed through the application of the Neural Architecture Search (NAS) technique, NASNet utilizes an automated process for designing neural networks, resulting in an architecture that is capable of providing superior performance compared to designs created by human experts[17]. A depiction of the NASNetMobile architecture's is presented in Figure 5. The white-colored input label signifies the concealed state from the input image. Meanwhile, the output, a pink hue, results from a concatenation operation encompassing all outcomes. Each convolutional cell emerges from B blocks, with a solitary block comprising two fundamental operations, denoted in yellow, and a conjoining operation marked in green. This intricate structure highlights the intricate interplay of primitive and combinatory operations within the architecture.

In this research, as documented in Tables 2 and 3, both architectures utilize the pre-trained from imagenet as the base model. The architecture is then extended with the addition of several layers. These layers are the flatten and dense layers. The flatten layer is used to convert the output of the previous layer into a one-dimensional vector, allowing the computational process in the next layer to be more efficient. Then, the dense layer is used to connect the results of the convolution process to pave the way for the classification process.

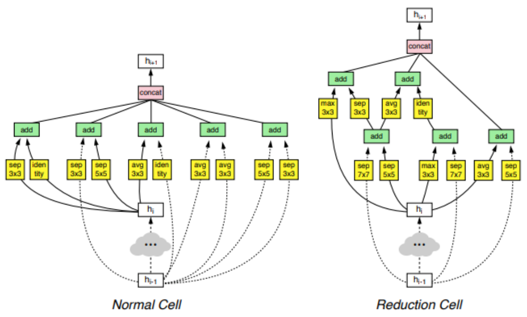

Figure 5. The Original Architecture of NASNetMobile

Table 2. The Architecture of MobileNetV2 in Research

Layer (type)	Output Shape
Input Layer	(None, 224, 224, 3)
MobileNetV2	(None, 7, 7, 1280)
flatten (Flatten)	(None, 62720)
dense (Dense)	(None, 256)
dense_1 (Dense)	(None, 75)

Table 3. The Architecture of NASNetMobile in Research

Layer (type)	Output Shape
Input Layer	**(None, 224, 224, 3)**
NASNetMobile	**(None, 7, 7, 1056)**
flatten (Flatten)	**(None, 62720)**
dense (Dense)	**(None, 256)**
dense_1 (Dense)	**(None, 75)**

In the dense layer, the RELU (Rectified Linear Activation) activation function is used to provide an element of non-linearity to the model (1). The Softmax activation function is also applied to the last dense layer to support the multi-class classification (2). Where X_m is the value of class m, while k is the number of classes and Euler number 2.71828.

$$Relu\ (i) = max(0, i) \tag{1}$$

$$softmax\ (x\_m) = \frac{e^{Xm}}{\sum_{n=1}^{k} e^{Xn}} \tag{2}$$

In addition, the optimization in this research uses Adam's algorithm. Adam's algorithm is a randomized stepwise optimization algorithm that combines the first and second moments of the gradient to update the parameters. Equations (3), (4), (5), (6), and (7) describe the Adam's Algorithm formula applied in this research. Where m is the momentum, beta is the parameter recurrence, the gradient is the gradient of the loss function for the parameter, p is the parameter, t is the iteration, lr is the learning rate, and epsilon is a small number to avoid division by zero.

$$m1 = beta1 \times m1 + (1 - beta1) \times gradient \tag{3}$$

$$m2 = beta2 \times m2 + (1 - beta2) \times gradient \tag{4}$$

$$m1_{hat} = m1 \div (1 - beta1^{t}) \tag{5}$$

$$m2_{hat} = m2 \div (1 - beta2^{t}) \tag{6}$$

$$p = p - lr \times m1_{hat} \div (\sqrt{m2_{hat}} + epsilon) \tag{7}$$

2.4. Evaluation

The use of evaluation metrics is necessary to determine the classification ability of the trained model. The evaluation metrics used in this research include accuracy, precision, and recall or sensitivity. Accuracy is the ratio of the correct prediction class to the total amount of data evaluated (8). Precision is the accuracy of correct positive predictions against all positive predictions (9). Recall or Sensitivity, measures the accuracy in making correct positive predictions against all correct predictions (10). Where TP, FP, TN, and FN are the number of cases classified as true positive, false positive, true negative, and false negative.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{8}$$

$$Precision = \frac{TP}{TP+FP} \tag{9}$$

$$Recall = \frac{TP}{TP+FN} \tag{10}$$

3. RESULT AND DISCUSSION

The implementation was carried out on the Kaggle platform, using a P-100 GPU with 16 GB of memory and 30 GB of RAM. Experiments were run in Python and supported by various libraries, such as TensorFlow, Keras, NumPy, scikit-learn, Matplotlib, and Pandas. The experimental process consists of two stages in the implementation phase. First, there is a training process with initial parameters, such as batch size 32, learning rate 0.001, epoch 25, and freeze on all layers. The second stage involves fine-tuning the architecture, which includes changing the batch size, and learning rate, and modifying the number of layers that are frozen.

Table 4 displays the implementation results of the proposed architectures, MobileNetV2 and NASNetMobile, on the datasets used in this research. A summary of the accuracy, precision, recall, and training time results for both architectures can be found in the table. MobileNetV2, using the default parameters, achieved an accuracy rate of 71%, with a training duration of 2 minutes and 3 seconds. Fine-tuning the parameters increased the accuracy to 86%, with a training time of 5 minutes and 26 seconds. In MobileNetV2 architecture tuning, optimal results were obtained by using a batch size of 16, a learning rate of 0.0001, and freezing the front 40% of the layer. On the other hand, NASNetMobile with default parameters produced an accuracy of 57%, with a training duration of 5 minutes and 3 seconds. However, with fine-tuning, the accuracy increased to 85%, and the training time took 10 minutes and 41 seconds. The tuning of the NASNetMobile architecture showed the best results with a batch size of 32, a learning rate of 0.0001, and freezing of the front 30% of the layer.

Both architectures showed almost identical precision and recall results when using fine-tuning parameters. However, significant differences were seen in the precision and recall results when using the default parameters. Although there were variations in the accuracy, precision, and recall values during training with the default parameters, the differences were not as pronounced when using the fine-tuning parameters. This shows that both architectures are capable of performing good classification for 75 classes of butterfly images.

Although there is a considerable difference in the accuracy levels before and after the tuning process, the training graph shows that the pattern of results does not substantially change between the pre- and post-tuning stages. In the MobileNetV2 architecture graph in Figure 6, there is an increase in accuracy at each epoch, although the stability does not reach the optimal level, both in the training accuracy graph and the validation accuracy graph.

Table 4. Implementation Result

Model	Accuracy	Precision		Recall		Training Time
		macro	weighted	macro	weighted	
MobileNetV2	0.7149	0.7505	0.7631	0.7175	0.7149	2 minutes 3 second
MobileNetV2 with fine-tuning	0.8677	0.8828	0.8865	0.8736	0.8677	5 minutes 26 seconds
NASNetMobile	0.5795	0.6391	0.6528	0.5725	0.5795	5 minutes 3 seconds
NASNetMobile with fine-tuning	0.8533	0.8669	0.8769	0.8552	0.8533	10 minutes 41 seconds

In addition, the pattern seen in the loss graph of the MobileNetV2 architecture shows significant improvement, with a decrease in loss values seen throughout the training and validation process as the epochs progress, although there is still instability. It is important to note that the loss graph of the MobileNetV2 architecture does not show any signs of overfitting, as the training loss graph and the validation loss graph remain relatively small differences. The results of the training graphs on the MobileNetV2 architecture show that the number of epochs used is appropriate for the desired conditions.

Similar to MobileNetV2, the NASNetMobile architecture shown in Figure 7 also shows an increase in accuracy at each epoch, and the stability does not reach the optimal level, both in the training accuracy graph and the validation accuracy graph.

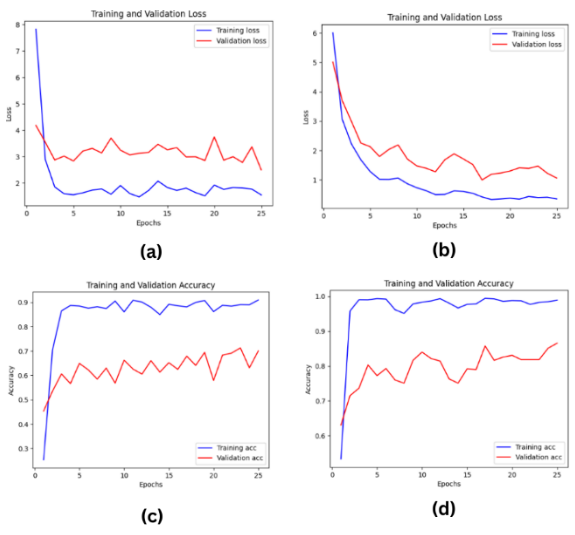

Figure 6. MobileNetV2 Graph (a) Loss Before Tuning (b) Loss After Tuning (c) Accuracy Before Tuning (d) Accuracy After Tuning

The pattern seen in the loss graph of the NASNetMobile architecture also indicates a significant improvement, with the loss value decreasing during the training and validation process as epochs pass, although there is still instability that needs to be considered. In the NASNetMobile architecture, as in MobileNetV2, there are no signs of overfitting, as the training loss graph and validation loss graph continue to show relatively small differences. Thus, the results of the training graph on the NASNetMobile architecture indicate that the number of epochs used is appropriate for the desired conditions.

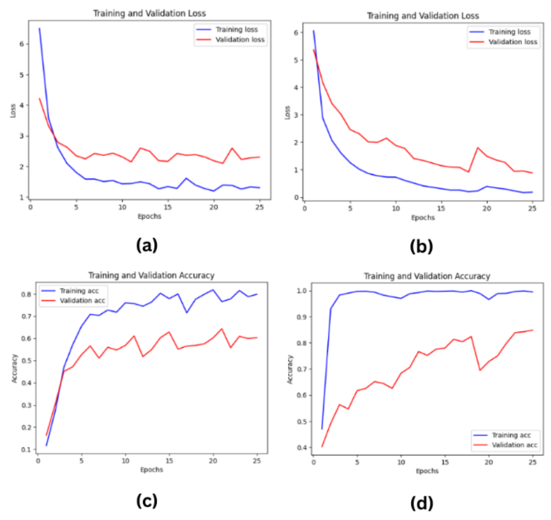

Figure 7. NASNetMobile Graph (a) Loss Before Tuning (b) Loss After Tuning (c) Accuracy Before Tuning (d) Accuracy After Tuning

4. DISCUSSION

This research evaluates the performance of two architectures, namely MobileNetV2 and

NASNetMobile, by applying the transfer learning method using pre-trained models from ImageNet as the foundation of the butterfly dataset. The experiments involved parameter adjustments and layer freezing for fine-tuning both architectures. MobileNetV2, when using the default parameters, achieved an accuracy rate of 71% in a training time of 2 minutes and 3 seconds. Through the application of fine-tuning the parameters, the accuracy increased to 86% but required a longer training time of 5 minutes and 26 seconds. In tuning the MobileNetV2 architecture, optimal results were achieved with a batch size of 16, a learning rate of 0.0001, and freezing of the front 40% of the layer. Meanwhile, NASNetMobile with default parameters initially recorded 57% accuracy in a training time of 5 minutes and 3 seconds. However, through fine-tuning, the accuracy improved to 85%, with a significantly longer training time of 10 minutes and 41 seconds. Tuning on the NASNetMobile architecture showed the best results with a batch size of 32, a learning rate of 0.0001, and freezing of the front 30% of the layer. Overall, the implementation results indicate that fine-tuning has a significant positive impact on improving model accuracy. From the results presented, the MobileNetV2 architecture shows superior accuracy performance compared to NASNetMobile on the dataset used.

The difference in accuracy values before and after the fine-tuning process is influenced by the parameter values during training and the number of layers frozen. The learning rate and batch size play a crucial role in influencing the parameters. Both architectures experienced optimal tuning at a learning rate of 0.0001. The application of a low learning rate has a positive impact on model convergence, especially when the training data has a high level of noise or variation, such as in this research, where the images in the dataset still contain backgrounds with various colors. Using a low learning rate allows the model to adjust parameters more carefully during the training process, reducing the risk of overfitting and improving overall convergence[18]. Batch size tuning also has a significant effect on the difference in accuracy levels before and after the fine-tuning process. On the MobileNetV2 architecture, the optimal batch size is 16, while on the NASNetMobile architecture, the optimal batch size reaches 32. Both batch sizes fall into the small category. The advantage of using a small batch size is seen in the ability of the model to converge more effectively[19]. By updating the parameters every iteration, the small batch size allows the model to respond responsively to data variations, making it easier to adjust to the specific characteristics of the dataset used. This provides greater flexibility in coping with data dynamics and diversity, as well as improving overall performance during the training process[20].

In addition, the number of layers that are fine-tuned also has a significant impact. In the context of this research, the fine-tuning process involves freezing many layers at the beginning of the architecture. This approach allows the model the flexibility to adapt to specific data while retaining the information obtained from the pre-trained model[21]. MobileNetV2 achieves optimal results when about 40% of all layers are freeze, while NASNetMobile achieves the best results with layer freezing at only the initial 30%. The impact of layer freezing occurs because Imagenet's pre-trained model already has an understanding of butterflies. However, to improve the performance in recognizing butterflies based on the dataset of this research, it is necessary to adjust the weights in the architecture. It is also important to note that the effect of layer freezing is manifested in the training time of the model[22]. The efficiency of training time, especially on the NASNetMobile architecture, takes longer. This is due to the larger number of weights being updated due to more layer freezes, as well as the larger number of layers built-in to NASNetMobile compared to MobileNetV2, which is about 5.3 million compared to 3.5 million[23].

In the discussion above, the utilization of transfer learning in MobileNetV2 and NASNetMobile has been proven to increase the accuracy rate in butterfly classification. The highest accuracy rate achieved was 86%, showing a significant improvement compared to previous studies that used the VGG19 architecture and only achieved an accuracy of 79.5%[12]. This study was also able to achieve a satisfactory accuracy rate by considering 75 different butterfly classes, a much larger number compared to previous studies that only focused on 4 and 10 butterfly classes[1], [12]. By utilizing transfer learning and a suitable fine-tuning process, this research can improve the accuracy of results even more and recognize many other types of butterflies.

5. CONCLUSION

Based on the evaluation, these two architectures show good ability in classifying butterfly images using the transfer learning method. Before the use of transfer learning, both architectures were tested using the default parameters of a learning rate of 0.001 and a batch size of 32. The accuracy result created by MobileNetV2 reached 71%, while NASNetMobile had a significant difference in accuracy of 57%. Furthermore, both architectures were trained with appropriate fine-tuning and the highest accuracy reached 86% by MobileNetV2 with a learning rate parameter of 0.0001, batch size of 16, and Adam optimizer. For NASNetMobile, the accuracy was slightly higher at 85% with a learning rate parameter of 0.0001, batch size of 32, and the same optimizer as MobileNetV2. The difference in results is due to the different architectural layers, the MobileNetV2 layer is lighter and the layer's understanding of the dataset is greater, which is 40% in the front layer, while NASNetMobile is only 30% in the front layer. In addition, the advantages of MobileNetV2 can be seen from the speed of pre-training, which takes 5 minutes and 26 seconds, unlike NasNetMobile which takes more time, 10 minutes and 41 seconds. This time difference is also caused by the number of layers in each architecture.

For future research to improve the accuracy rate, a segmentation approach can be implemented. This method involves applying segmentation techniques such as K-Means and GrabCut, or even utilizing deep learning approaches such as U2Net. In addition, to balance the distribution of the number of datasets in each class, augmentation or undersampling measures can be taken. These efforts are expected to produce more optimized results and improve the overall accuracy of the research.

BIBLIOGRAPHY

[1] N. N. K. Arzar, N. Sabri, N. F. M. Johari, A. A. Shari, M. R. M. Noordin, and S. Ibrahim, "IEEE International Conference on Automatic Control and Intelligent Systems," in *IEEE Control Systems Society. Chapter MalaysiaInstitute of Electrical and Electronics Engineers*, 2019.

[2] E. Hartati, K. Kunci, and K. Kupu, "Klasifikasi Spesies Kupu Kupu Menggunakan Metode Convolutional Neural Network," in *MDP Student Conference (MSC)*, 2022, pp. 569–577.

[3] L. Zhu and P. Spachos, "Towards Image Classification with Machine Learning Methodologies for Smartphones," *Mach Learn Knowl Extr*, vol. 1, no. 4, pp. 1039–1057, Dec. 2019, doi: 10.3390/make1040059.

[4] T. Y. Chen, "MonarchNet: Differentiating Monarch Butterflies from Butterflies Species with Similar Phenotypes," Jan. 2022, doi: 10.1096/fasebj.2021.35.S1.05504.

[5] B. A. Bakri, Z. Ahmad, and S. M. Hatim, "Butterfly family detection and identification using convolutional neural network for lepidopterology," *International Journal of Recent Technology and Engineering*, vol. 8, no. 2 Special Issue 11, pp. 635–640, Sep. 2019, doi: 10.35940/ijrte.B1099.0982S1119.

[6] H. He, "The Comparison and Analysis of Classic Convolutional Neural Network in the Field of Computer Vision," in *IOP Conference Series: Materials Science and Engineering*, Institute of Physics Publishing, Mar. 2020. doi: 10.1088/1757-899X/740/1/012153.

[7] A. K. Sharma *et al.*, "Dermatologist-Level Classification of Skin Cancer Using Cascaded Ensembling of Convolutional Neural Network and Handcrafted Features Based Deep Neural Network," *IEEE Access*, vol. 10, pp. 17920–17932, 2022, doi: 10.1109/ACCESS.2022.3149824.

[8] P. Langgeng, W. E. Putra, M. Naufal, and E. Y. Hidayat, "A Comparative Study of MobileNet Architecture Optimizer for Crowd Prediction," *Semarang 123 Jl. Imam Bonjol No*, vol. 8, no. 3, p. 50131, 2023.

[9] T. Ridnik, E. Ben-Baruch, A. Noy, and L. Zelnik-Manor, "ImageNet-21K Pretraining for the Masses," Apr. 2021, [Online]. Available: http://arxiv.org/abs/2104.10972

[10] Livinus Ifunanya Umeaduma, "Survey of image classification models for transfer learning," *World Journal of Advanced Research and Reviews*, vol. 21, no. 1, pp. 373–383, Jan. 2024, doi: 10.30574/wjarr.2024.21.1.0006.

[11] A. S. C. K. Ravi Joshi Annapurna Maritammanavar, "Transfer Learning in Computer Vision: Technique and applications," *Tuijin Jishu/Journal of Propulsion Technology*, 2023, [Online]. Available: https://api.semanticscholar.org/CorpusID:265648825

[12] A. S. Almryad and H. Kutucu, "Automatic identification for field butterflies by convolutional neural networks," *Engineering Science and Technology, an International Journal*, vol. 23, no. 1, pp. 189–195, Feb. 2020, doi: 10.1016/j.jestch.2020.01.006.

[13] T. S. Alves *et al.*, "Automatic detection and classification of honey bee comb cells using deep learning," *Comput Electron Agric*, vol. 170, Mar. 2020, doi: 10.1016/j.compag.2020.105244.

[14] P. Anantha Prabha, G. Suchitra, and R. Saravanan, "Cephalopods Classification Using Fine Tuned Lightweight Transfer Learning Models," *Intelligent Automation and Soft Computing*, vol. 35, no. 3, pp. 3065–3079, 2023, doi: 10.32604/iasc.2023.030017.

[15] K. Apivanichkul, P. Phasukkit, and P. Dankulchai, "The Effect of Preprocessing on U-Net for Bladder Segmentation in CT Images," in *International STEM Education Conference (iSTEM-Ed)*, 2023, pp. 1–5. doi: 10.1109/iSTEM-Ed59413.2023.10305805.

[16] DIPIE, "https://www.kaggle.com/datasets/phucthaiv02/butterfly-image-classification," KAGGLE.

[17] H. K. Dishar and L. A. Muhammed, "Detection Brain Tumor Disease Using a Combination of Xception and NASNetMobile," *International Journal of Advances in Soft Computing and its*

Applications, vol. 15, no. 2, pp. 325–336, 2023, doi: 10.15849/IJASCA.230720.22.

[18] Y. Ding, "The Impact of Learning Rate Decay and Periodical Learning Rate Restart on Artificial Neural Network," in *Proceedings of the 2021 2nd International Conference on Artificial Intelligence in Electronics Engineering*, in AIEE '21. New York, NY, USA: Association for Computing Machinery, 2021, pp. 6–14. doi: 10.1145/3460268.3460270.

[19] H. Iiduka, "The Number of Steps Needed for Nonconvex Optimization of a Deep Learning Optimizer is a Rational Function of Batch Size," Aug. 2021, [Online]. Available: http://arxiv.org/abs/2108.11713

[20] Z. Hu, J. Xiao, N. Sun, and G. Tan, "Fast and accurate variable batch size convolution neural network training on large scale distributed systems," *Concurr Comput*, vol. 34, no. 21, p. e7119, 2022, doi: https://doi.org/10.1002/cpe.7119.

[21] R. Anditto and R. Roestam, "SECURITY MONITORING USING IMPROVED MOBILENET V2 WITH FINE-TUNING TO PREVENT THEFT IN RESIDENTIAL AREAS DURING THE COVID-19 PANDEMIC," *Science and Information Technology*, vol. Vol. 5, no. No 1, pp. 87–94, 2022, [Online]. Available: https://doi.org/10.31598

[22] Q. Li, J. Zhang, and Z. Chen, "Detection on difficult small objects using layer-wise training strategy," in *Proceedings of 2020 IEEE 3rd International Conference of Safe Production and Informatization, IICSPI 2020*, Institute of Electrical and Electronics Engineers Inc., Nov. 2020, pp. 282–286. doi: 10.1109/IICSPI51290.2020.9332370.

[23] T. Hong Chun *et al.*, "Efficacy of the Image Augmentation Method using CNN Transfer Learning in Identification of Timber Defect," 2022. [Online]. Available: www.ijacsa.thesai.org.