Jurnal Elektronika dan Telekomunikasi (JET), Vol. 22, No. 2, December 2022, pp. 87-94
Accredited by KEMDIKBUDRISTEK, Decree No: **158/E/KPT/2021**
doi: 10.55981/jet.503

Comparison of Classification of Birds Using Lightweight Deep Convolutional Neural Networks

Aldi Jakaria[a, *], Hilman Ferdinandus Pardede[a, b]

[a]*Fakultas teknologi informasi*
Universitas Nusa Mandiri
Jl. Kramat Raya No.18, RW.7, Kwitang, Kec. Senen
Jakarta, Indonesia
[b]*Research Center for Data and Information Sciences*
National Research and Innovation Agency
Komplek BRIN Jl. Sangkuriang No. 21
Bandung, Indonesia

Abstract

Environmental scientists often use birds to understand ecosystems because they are sensitive to environmental changes, but few experts are available. To make it easier to recognize bird species, an automatic system that can classify bird species is needed. There are lots of models to choose from, but some models require very high computational data when training data, reducing training time can result in less wasted electrical energy so that it can have a good effect on the environment. For this reason, it is necessary to test a model that has a small complexity in training time, whether it can produce good performance. Based on the number of neural network models available, this study will classify using the EfficientNet, EfficientNetV2, MobileNet, MobileNetV2, and NasnetMobile models to determine whether these models can have good performance. From the research results, all the models tested have good performance with an accuracy between 95% - 97%. The MobileNetV2 model has the less efficiency with the smallest training time while maintaining good performance.

Keywords: EfficientNet, EfficientNetV2, MobileNet, MobileNetV2, NasnetMobile.

I. INTRODUCTION

Birds are very important in maintaining the balance of the ecosystem. Studying birds can help to better understand the world around us and to understand important information about nature. Bird identification is a well-known problem for ornithologists. Environmental scientists often use birds to understand ecosystems because they are sensitive to environmental changes. Various real-world applications rely on birds, such as environmental pollution monitoring [1].

The presence of birds of different species in an ecosystem is also important for many environmental reasons. This is another area where our classification program can be useful [2]. With the help of our classification method, authorities can also track bird hunting in an area by monitoring the rise and fall of the population of each bird species [3].

Identification of bird species is very difficult because of the large number of bird species, so an expert is needed. Unfortunately the number of experts available is limited [4]. To make it easier to recognize and study bird species, a system is needed to be able to automatically detect bird species, one of which is image identification. In this study, we will discuss more specifically on the identification of bird species [5].

Image classification tends to have very high computational requirements. There are lots of models to choose from, but some models require very high computations both when conducting data training, reducing training time can result in less wasted electrical energy [6].

During the training of machine learning, computers need so many electrical powers to the strain on power plants to produce enough energy. The production of energy creates pollution and emissions. The amount of electricity needed to power computers contributes to the millions of tons of greenhouse gases that are emitted into the atmosphere each year [7].

In other cases, certain models produce very many parameters. By looking for an efficient model we can reduce the amount of computing very much so that the existing model can be run on more devices that have less computing power such as mobile devices and internet of things devices [8].

One of the most popular models is the Convolutional Neural Network (CNN) which was introduced by LeChun [9]. However, this model does not have very good performance, the training time is very long due to the small number of parameters and training must be carried out from the beginning. In addition, there is a Visual Geometry Group (VGG) [10] model which has good performance, but makes the number of parameters large, and hence increasing the training time. Thus, increasing the complexity of the model.

There are several models available that focus on model efficiency. One of the models that can be used is the EfficientNet model created in 2019, EfficientNet is a systematic model, identifying carefully to consider the depth, network width, and resolution that can lead to

* Corresponding Author.
Email: 14002457@nusamandiri.ac.id
Received: September 09, 2022 ; Revised: December 15, 2022
Accepted: December 16, 2022 ; Published: December 31, 2022

better performance. Representation of the combination of efficient and good accuracy with varying degrees of scale, this architecture gave birth to models from B0 to B7 [11].

In 2021, the latest version of EfficientNet, namely EfficientNetV2, has a smaller model size and results in shorter training times. A combination of training-aware neural architecture search (NAS) and scaling is used, to optimize training speed and parameter efficiency. The EfficientNetV2 model requires much faster training time than the previous version with a size up to 6.8 times smaller [12].

In addition, there is also the MobileNetV1 model introduced by Google in 2017 which offers a model that has few parameters, and small complexity. This model uses Depthwise Separable Convolution to reduce the size complexity. This is especially useful for mobile applications and embedded vision applications [13].

The development of MobileNetV1, namely MobileNetV2, was introduced by Google in 2019. In this model, a better module was introduced with an inverted residual structure. The non-linearity in the narrow layers is removed this time. With MobileNetV2 as a tool for feature extraction, good performance can also be achieved for object detection and image segmentation [12].

In 2018 Google Brain introduced the Nasnet model. In this model, it is proposed to find architectural building blocks in a small data set and then transfer those blocks to a larger data set. In addition, simply by varying the number of convoluted cells and the number of filters in the convoluted cells, different versions of NASNet can be created with different computational requirements. Thanks to this property of cells, it is possible to produce a set of models that achieve better accuracy than all the models created at the time with an equivalent or smaller computational budget [14].

Based on the number of neural network models available, this study will classify using the EfficientNet, EfficientNetV2, MobileNet, MobileNetV2, and NasnetMobile models to determine whether these models can have good performance. These models were chosen because they are models that have a small number of parameters. Then look for the optimal configuration so that it can get a model that has a small complexity and training time that is not too long, without sacrificing performance.

In this paper, we contribute on evaluating several lightweight deep convolutional neural networks for birds' classifications. To the best of our knowledge, evaluations on lightweights CNN only for this dataset has not been performed previously. By doing so, we could find most suitable lightweight models for this dataset.

II. METHOD

Figure 1 is the method that we will be using in this experiment.

A. Dataset Generation

In this step, data is selected from a set of operational data. This paper uses data on 300 bird species. 47332 training images, 1625 test images (5 images per species) and 1625 validation images (5 images per species). All images are 224×224×3 color images in jpg format. Data cleaning is carried out including deleting data duplication, checking data inconsistencies, and data errors. After selecting the data and pre-processing the data, the next step is to transform the data. The data in the form of images is converted into a matrix form by using ImageDataGenerator.

B. Model Selection

This study uses a neural network model to process data, namely MobileNet, MobileNetV2, NasnetMobile, EfficientNetB0, and EfficientNetV2B0 models. For each model, 3 variations are carried out, namely:

a. Without Transfer Learning
b. With Transfer Learning and all Layers is Freeze
c. With Transfer Learning dan all Layers is not Freeze.

The reason why the experiment is divided is to find the balance between performance and complexity across all models so that we can determine the best configuration for the model.

1) MobileNetV1

The structure of MobileNet is built on Depthwise Separable Convolution in depth as mentioned in the previous section except for the first layer which is full convolution. By defining the network in such simple terms, we can easily explore the network topology to find a good network. The MobileNet architecture is defined in Figure 2.2. All layers were followed by batchnorm and ReLU nonlinearity with the exception of the fully connected final layer which had no nonlinearity and was included in the softmax layer for classification. Figure 2.1 contrasts layers with regular, batchnorm, and nonlinear ReLU convolutions to factored tar layers with deep convolutions, 1×1 pointwise convolution as well as batchnorma and ReLU after each layer convolution. Bottom sampling is handled with strided convolutions in depthwise convolutions as well as in the first layer. The final mean pool reduces the spatial resolution to 1 before the fully connected layers. Counting the Depthwise and pointwise convoys as separate layers, MobileNet has 28 layers [13].

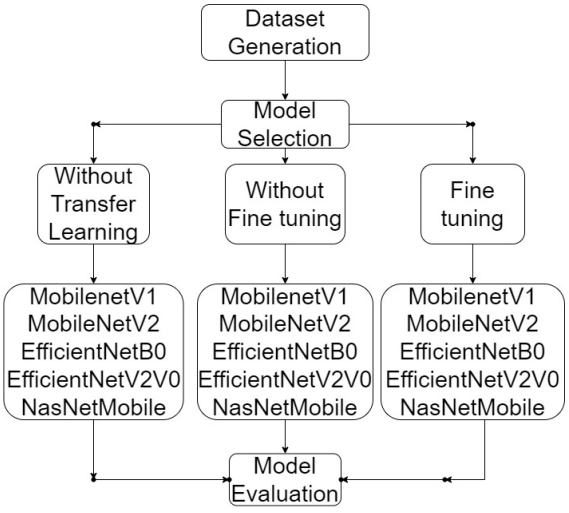

Figure 1. Research method.

2) MobileNetV2

The MobileNetV2 architecture consists of an initial full convolutional layer with 32 filters, which is followed by 19 residual bottleneck layers. It utilizes ReLU6 as the non-linear activation function due to its reliability when used with low-precision computing. The network employs 3×3 kernels and includes dropout and batch normalization during training. Except for the first layer, it uses a constant expansion rate throughout the network. In experiments, it was found that expansion rates between 5 and 10 yield nearly identical performance curves, with smaller networks getting better with slightly smaller expansion rates and larger networks performing slightly better with larger expansion rates. For all main experiments, an expansion factor of 6 was applied to the size of the input tensor. For example, for a bottleneck layer that takes an input tensor of 64 channels and produces a tensor with 128 channels, the intermediate expansion layer then becomes 64 6 = 384 channels [13].

3) EfficientNet

The main building block of this network consists of MBConv which is added with squeeze and excitation optimizations. MBConv is similar to the inverted residual block used in MobileNetV2. This forms a shortcut connection between the start and end of the convolution block. The input activation map is first expanded using 1x1 convolution to increase the depth of the feature map. This is followed by 3×3 Depthwise convolutions and Point-wise convolutions which reduce the number of channels in the output feature map. Shortcut joints connect the narrow layers while wider layers are present between the skip joints. This structure helps in reducing the overall number of operations required as well as the size of the model [15].

4) EfficientNetV2

Compared to the EfficientNet backbone, the EfficientNetV2 we are looking for has a few key differences:

a. The first difference is that EfficientNetV2 extensively uses the newly added MBConv MBConv in the initial layer.
b. Second, EfficientNetV2 prefers a smaller expansion ratio for MBConv, because a smaller expansion ratio tends to have less memory access overhead.
c. Third, EfficientNetV2 prefers a smaller 3×3 kernel size, but adds more layers to compensate for the reduction in receptive fields resulting from the smaller kernel size.
d. Lastly, EfficientNetV2 completely removed the last step-1 stage in the original EfficientNet, probably due to its large parameter size and memory access overhead [16].

5) NasnetMobile

The primary search method we use in this work is the Neural Architecture Search (NAS) framework. In NAS, the iterative neural network (RNN) controller samples child networks with different architectures. The child network is trained for convergence to obtain accuracy on the pending validation set. The resulting accuracy is used to update the controller so that the controller will produce a better architecture over time. Controller weights are updated with policy gradients [14].

Figure 2. Birds image sample.

C. Model Evaluation

The resulting data is then analyzed to find the model with the least complexity while maintaining good performance.

III. RESULTS

A. Dataset Generation

The data that used in this work is obtained from Kaggle Birds 450 Species - Image Classification (previously 300 species image). Images were gathered from internet searches by species name. Once the image files for a species was downloaded, they were checked for duplicate images using our developed python duplicate image detector. All duplicate images detected were deleted in order to prevent their being images common between the training, test and validation sets. The data set consists of 300 bird species, 47332 training images, 1625 test images and 1625 validation images (5 images per species. All images are 224×224×3 color images in jpg format).

Figure 2 is an example of a sample image of a bird species that already has a label. The images were collected from an internet search by species name. After the image files for a species are downloaded, they are checked for duplicate image. All detected duplicate images were removed to prevent them from becoming common images among training and validation sets.

After that, the image is then cropped so that the bird occupies at least 50% of the pixels in the image. Then the image is resized to 224×224×3 in jpg format. Image cropping is done, so that, at the time of training, the image has sufficient information to make an accurate classifier. All files are also numbered sequentially starting from one for each species. So, the test images are named 001.jpg to 005.jpg. Similarly for the validation image. The null prefix is preserved in the order of the files when used to make the transformation process easier in Python.

The training set is unbalanced, having varying number of files per species. However, each species has at least 120 training image files. One of the significant imbalances in the data set is the ratio of male species images to female species images. About 85% of the images are male and 15% are female. Males are characteristically far more diverse in color than females. As a result, the images of males and females may look very different. Almost all training and validation images

were taken from males of that species. As a result, classifiers may not work well on images of female species.

Then the image is processed using Python and Jupyter notebook tools. The following are the computer specifications used in this study:

1. Intel core i7 9750H
2. Ram 32GB DDR4
3. Nvidia Rtx 2060 Mobile
4. SSD NVMe 1 TB

The collected image is then transformed into a matrix form using ImageDataGenerator from the Keras library. With the following conditions:

1. Rescale = 1/255
2. Shear range = 0.2
3. Zoom range = 0.2
4. Horizontal flip = true
5. Class mode = categorical

There are 43622 training data images consisting of 300 target classes and 1500 validation data images with 300 target classes. All training was carried out using the Adam optimizer and loss function categorical cross entropy and epoch 100, however, an early stopping configuration was carried out which monitored the accuracy metric with a delta value of 0.001 and a tolerance of 3 epochs. So, if for 3 epochs there is no minimum accuracy increase of 0.001, then the training process will stop. The results of the training are then displaying accuracy metrics and accuracy validation in the form of graphs, besides that a confusion matrix is also displayed to perform further analysis.

B. Model Selection

The model prepared is a model that has a small complexity that is available in Keras.

1) MobileNetV1

a) Without Transfer Learning

Figure 3 contains the result of MobileNetV1 without transfer learning. With an input size of 112×112, the learning rate of 0.0001 stops at the 59th epoch, the total training duration is 2 hours 52 minutes. The resulting accuracy reaches 97% and validation accuracy is 86%.

b) With Transfer Learning and All Layers are Frozen

Figure 4 contains the result of MobileNetV1 with transfer learning where all layers are frozen. With an input size of 112×112, the learning rate of 0.0001 stops at the 57th epoch, the total training duration is 2 hours and 38 minutes. The resulting accuracy reaches 94% and validation accuracy is 93%.

c) With Transfer Learning dan All Layers are Not Frozen

Figure 5 contains the result of MobileNetV1 with transfer learning where all layers are not frozen. With an input size of 112×112, the learning rate of 0.0001 stops at the 27th epoch, the total training duration is 1 hour 17 minutes. The resulting accuracy reaches 97% and validation accuracy reaches 96%.

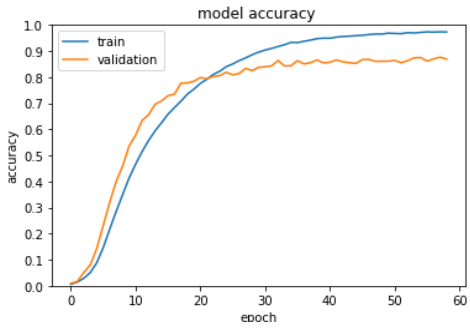

Figure 3. MobileNetV1 without transfer learning.

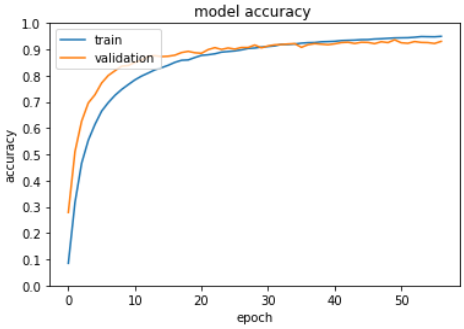

Figure 4. MobileNetV2 with transfer learning and all layers are frozen.

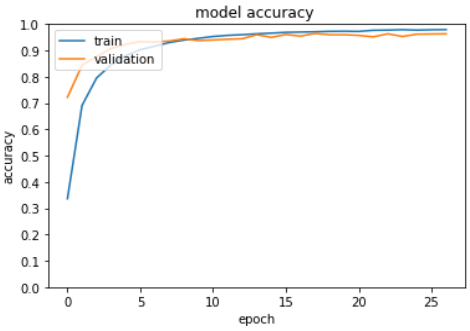

Figure 5. MobileNetV1 with transfer learning and all layers are not frozen.

2) MobileNetV2

a) Without Transfer Learning

Figure 6 contains the result of MobileNetV2 without transfer learning. With an input size of 112×112, the learning rate of 0.0001 stops at the 58th epoch, the total training duration is 2 hours 49 minutes. The resulting accuracy reaches 96% and validation accuracy reaches 86%.

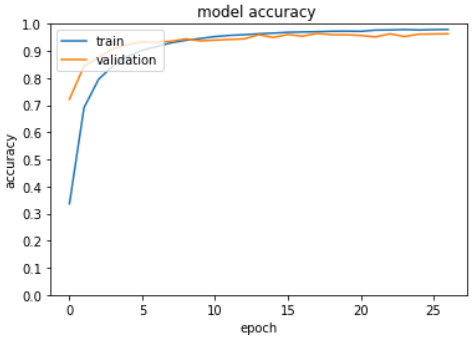

Figure 6. NasNetMobile without transfer learning.

b) With Transfer Learning and All Layers are Frozen

Figure 7 contains the result of MobileNetV2 with transfer learning where all layers are frozen. With an input size of 112×112, the learning rate of 0.0001 stops at the 55th epoch, the total training duration is 2 hours 29 minutes. The resulting accuracy reaches 96% and validation accuracy reaches 94%.

c) With Transfer Learning and All Layers are Not Frozen

Figure 8 contains the result of MobileNetV2 with transfer learning where all layers are not frozen. With an input size of 112×112, the learning rate of 0.0001 stops at the 25th epoch, the total training duration is 1 hour 12 minutes. The resulting accuracy reaches 97% and validation accuracy is 96%.

3) NasNetMobile

a) Without Transfer Learning

Figure 9 contains the result of NasNetMobile without transfer learning. With an input size of 112×112, the learning rate of 0.0001 stops at the 50th epoch, the total training duration is 3 hours and 15 minutes. The resulting accuracy reaches 95% and validation accuracy is 79%.

b) With Transfer Learning and All Layers are Frozen

Figure 10 contains the result of NasNetMobile with transfer learning and all layers are frozen. With an input size of 224×224, the learning rate of 0.0001 stops at the 42nd epoch, the total training duration is 6 hours 18 minutes. The resulting accuracy reaches 97% and validation accuracy reaches 97%.

c) With Transfer Learning and All Layers are Not Frozen

Figure 11 contains the result of NasNetMobile with transfer learning and all layers are not frozen. With an input size of 224×224, the learning rate of 0.0001 stops at the 20th epoch, the total training duration is 3 hours and 12 minutes. The resulting accuracy reaches 98% and validation accuracy is 97%.

4) EfficientNetB0

a) Without Transfer Learning

Figure 12 contains the result of EfficientNetB0 without transfer learning. With an input size of 112×112, the learning rate of 0.0001 stops at the 72nd epoch, the total training duration is 3 hours and 34 minutes. The resulting accuracy reaches 96% and validation accuracy is 87%.

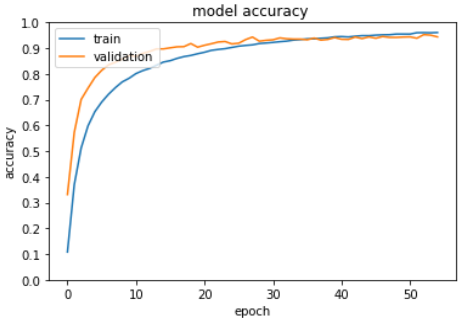

Figure 7. MobileNetV2 with transfer learning and all layers are frozen.

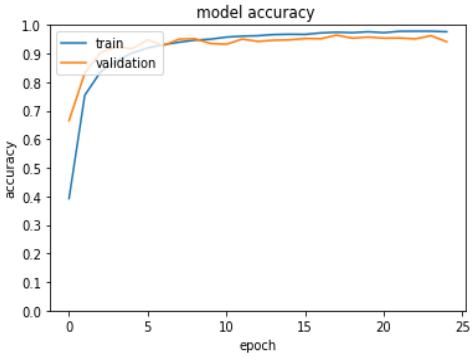

Figure 8. MobileNetV2 with transfer learning and all layers are not frozen.

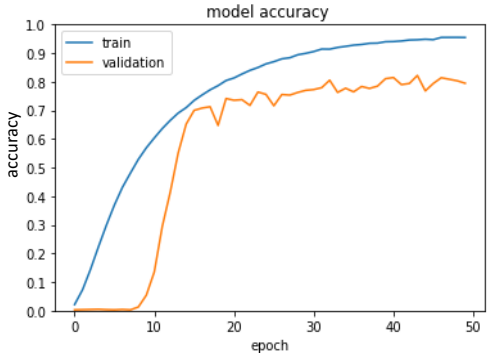

Figure 9. NasNetMobile without transfer learning.

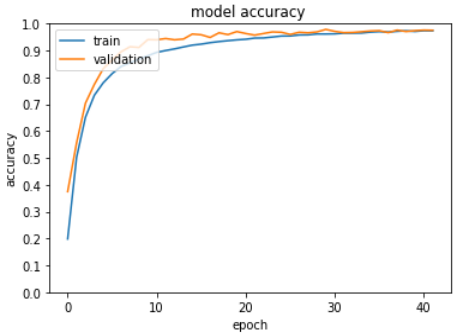

Figure 10. NasNetMobile with transfer learning and all layers are frozen.

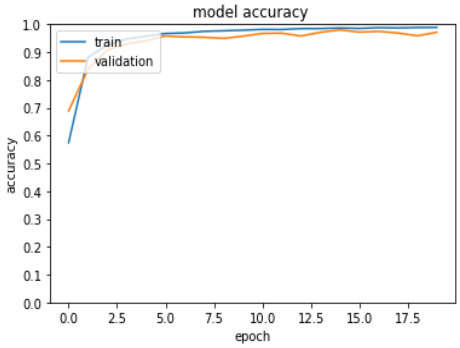

Figure 11. NasNetMobile with transfer learning and all layers are not frozen.

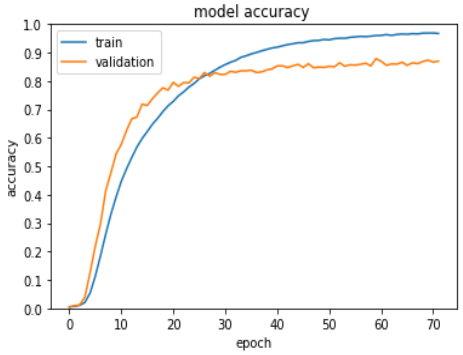

Figure 12. EfficientNetB0 without transfer learning.

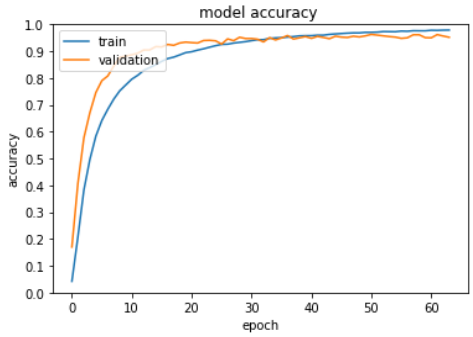

Figure 13. EfficientNetB0 with transfer learning and all layers are frozen.

b) With Transfer Learning and All Layers are Frozen

Figure 13 contains the result of EfficientNetB0 with transfer learning where all layers are frozen. With an input size of 112×112, the learning rate of 0.0001 stops at the 64th epoch, the total training duration is 3 hours and 3 minutes. The resulting accuracy reaches 97% and the validation accuracy is 95%.

c) With Transfer Learning and All Layers are Not Frozen

Figure 14 contains the result of EfficientNetB0 with transfer learning where all layers are not frozen. With an input size of 112×112, the learning rate of 0.00001 stops at the 63rd epoch, the total training duration is 3 hours 5 minutes. The resulting accuracy reaches 98% and validation accuracy is 97%.

5) EfficientNetV2B0

a) Without Transfer Learning

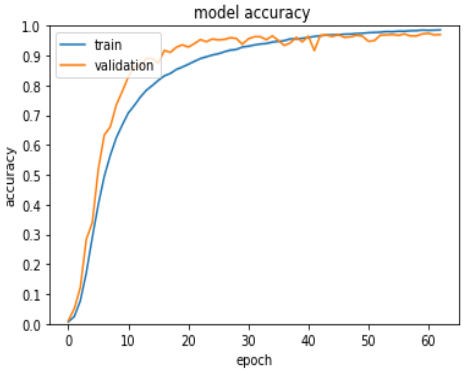

Figure 14. EfficientNetB0 with transfer learning and all layers are not frozen.

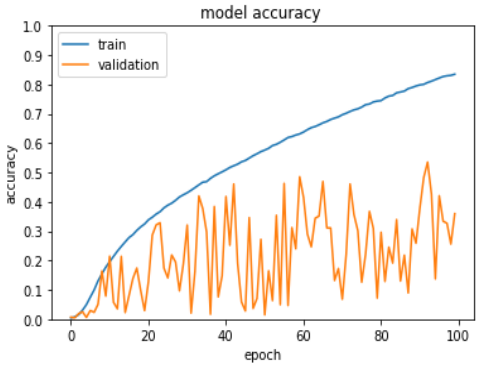

Figure 15. EfficientNetV2B0 without transfer learning.

Figure 15 contains the result of EfficientNetV2B0 without transfer learning. With an input size of 112×112, the learning rate of 0.00001 stops at the 100th epoch, the total training duration is 4 hours and 54 minutes. The resulting accuracy reaches 83% and validation accuracy is 35%.

b) With Transfer Learning and All Layers are Frozen

Figure 16 contains the result of EfficientNetV2B0 with transfer learning where all layers are frozen. With an input size of 112×112, the learning rate of 0.0001 stops at the 66th epoch, the total training duration is 2 hours 59 minutes. The resulting accuracy reaches 97% and validation accuracy is 93%.

c) With Transfer Learning dan All Layers are Not Frozen

Figure 17 contains the result of EfficientNetV2B0 with transfer learning where all layers are not frozen. With an input size of 112×112, the learning rate of

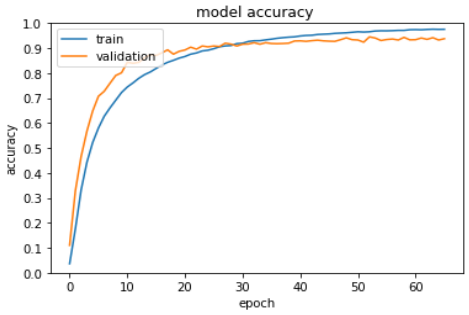

Figure 16. EfficientNetV2B0 with transfer learning and all layers are frozen.

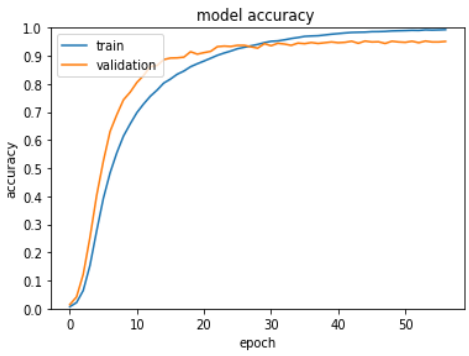

Figure 17. EfficientNetV2B0 with transfer learning and all layers are not frozen.

0.00001 stops at the 57th epoch, the total training duration is 3 hours 6 minutes. The resulting accuracy reaches 99% and the validation accuracy is 95%.

C. Model Evaluation

1) Without Transfer Learning

Table 1 contains the result of the models without transfer learning.

2) With Transfer Learning and All Layers are Frozen

Table 2 contains the result of the models with transfer learning and all layers are frozen.

3) With Transfer Learning and All Layers are Not Frozen

Table 3 contains the result of the models with transfer learning and all layers are not frozen.

IV. Discussion

We compare the complexity of the model based on the size of the model and the training time. Based on the experimental results, the MobileNetV2 transfer learning model with fine tuning has the less training time for optimization.

Based on the results analysis, it can be seen that the MobileNetV2 transfer learning model with fine tuning has the lowest training time. It was also found that each variation model without transfer learning tends to require a longer training time. The deficiency of the MobileNetV2 model is that the accuracy is slightly smaller than other models.

There are a few reasons that make MobileNet architecture require less training time despite still maintaining good performance which dramatically reduce the complexity cost and model size of the network, which is suitable to Mobile devices, or any devices with low computational power. In MobileNetV2, a better module is introduced with inverted residual structure. Non-linearities in narrow layers are removed this time. With MobileNetV2 as backbone for feature extraction, state-of-the-art performances are also achieved for object detection and semantic segmentation.

One important consideration when using the MobileNetV2 transfer learning model with fine tuning is the availability of high-quality training data. In order to learn effectively and improve its performance, it is necessary to provide a large and diverse set of training examples in the model. This can be challenging, especially for specialized tasks or datasets, but it is essential for achieving good results with the model.

Another important factor to consider is the hyperparameters used for training the model. These include the learning rate, batch size, and other parameters that can affect the model's performance and convergence during training. Tuning these hyperparameters can be time-consuming, but it can help improve the model's accuracy and efficiency.

Overall, the MobileNetV2 transfer learning model with fine tuning offers a number of benefits for many applications. It can provide good performance and efficiency, especially on mobile devices, and it can be easily fine-tuned to improve its accuracy and performance on specific tasks or datasets. Further research and exploration of this model, as well as other transfer learning techniques, can help improve its capabilities and applications.

In future research, it would be interesting to explore the use of the MobileNetV3 model, which is the latest version of the MobileNet architecture. This model is designed to offer even better performance and efficiency compared to MobileNetV2, and it could potentially be used in a wider range of applications. Additionally, further research could focus on fine-tuning the MobileNetV3 model to improve its accuracy and performance on specific tasks or datasets.

Table 1
Model Comparison Without Transfer Learning

Model	Acc	Val Acc	Epoch	Duration	Size (MB)
MobileNet	0,9734	0,8693	59	2:52:04	41.322
Mobile NetV2	0,9659	0,8620	58	2:49:47	30.799
Nasnet Mobile	0,9547	0,7947	50	3:15:36	53.795
Efficient NetB0	0,9675	0,8700	72	3:34:00	51.759
Efficient NetV2B0	0,8349	0,3593	100	4:54:10	20.739

Table 2
Model Comparison with Transfer Learning where All Layers are Frozen

Model	Acc	Val Acc	Epoch	Duration	Size (MB)
MobileNet	0,9499	0,9307	57	2:38:17	16,429
Mobile NetV2	0,9609	0,9440	55	2:29:42	13,673
Nasnet Mobile	0,9736	0,9753	42	6:18:52	21,046
Efficient NetB0	0,9789	0,9520	64	3:03:27	20,739
Efficient NetV2B0	0,9756	0,9380	66	2:59:30	28,202

Table 3
Model Comparison with Transfer Learning where All Layers are Not Frozen

Model	Acc	Val Acc	Epoch	Duration	Size (MB)
MobileNet	0,9794	0,9633	27	1:17:28	41,322
Mobile NetV2	0,9766	0,9413	25	1:12:30	30,799
Nasnet Mobile	0,9889	0,9713	20	3:12:18	53,795
Efficient NetB0	0,9860	0,9707	63	3:05:33	51,759
Efficient NetV2B0	0,9932	0,9513	57	3:06:33	73,543

V. Conclusion

This paper compared various models of lightweight deep convolutional neural networks for birds' classification. Models that have small sizes such as EfficientNet, EfficientNetV2, MobileNet, MobileNetV2, and NasnetMobile are able to classify 300 bird species and have good performance without reducing performance with an accuracy of 95-97% validation data.

The most optimal model configuration is the MobileNetV2 With an input size of 112×112, the learning rate of 0.0001 stops at the 25th epoch, which requires the least training time of 1 hour 12 minutes. The models need a significant smaller amount of time to train, which will use less electricity and less impact to the environment.

DECLARATIONS

Conflict of Interest

The authors have declared that no competing interests exist.

CRediT Authorship Contribution

Aldi Jakaria: Software, Data Curation, Writing - Original Draft Preparation, Visualization, Writing - Review & Editing; Hilman Ferdinandus Pardede: Conceptualization, Methodology, Investigation, and Supervision.

Funding

The authors received no financial support for the publication of this article.

REFERENCES

[1] Y. Rachmawati, Y. W. N. Tri, and A. P. Milenia, "Keanekaragaman jenis aves dan status konservasi di area pemandian air panas cangar , jawa timur 2019," (in Indonesia), in *Seminar Nasional Pendidikan Biologi dan Saintek (SNPBS) ke-4*, Jul. 2019 pp. 436–444, 2019. [Online]. Available: https://publikasiilmiah.ums.ac.id/handle/11617/11355

[2] P. Gavali and J. S. Banu, "Bird species identification using deep learning on gpu platform," in *Int. Conf. Emerg. Trends Inf. Technol. Eng. ic-ETITE 2020*, Vellore, India, pp. 1–6, 2020, doi: 10.1109/ic-ETITE47903.2020.85.

[3] S. Islam, S. I. A. Khan, M. Minhazul Abedin, K. M. Habibullah, and A. K. Das, "Bird species classification from an image using vgg-16 network," in *ACM Int. Conf. Proceeding Ser.*, Bangkok, Thailand, pp. 38–42, 2019, doi: 10.1145/3348445.3348480.

[4] S. Kumar, V. Dhoundiyal, N. Raj, and N. Sharma, "A comparison of different techniques used for classification of bird species from images," in Smart and Sustainable Intelligent Systems, N. Gupta, P. Chatterjee and T. Choudhury, Eds., Beverly, USA: Scrivener Publishing LLC, 2021, pp. 41–50, doi: 10.1002/9781119752134.ch3.

[5] G. Gupta, M. Kshirsagar, M. Zhong, S. Gholami, and J. L. Ferres, "Comparing recurrent convolutional neural networks for large scale bird species classification," *Sci. Rep.*, vol. 11, no. 1, pp. 1–12, 2021, doi: 10.1038/s41598-021-96446-w.

[6] S. Hartati, "Kecerdasan Buatan Berbasis Pengetahuan.", 1st ed. Yogyakarta, Indonesia: UGM Press, 2021. [Online]. Available: https://books.google.co.id/books?id=cnlREAAAQBAJ.

[7] N. I. Sarkar and S. Gul, "Green computing and internet of things for smart cities: technologies, challenges, and implementation," in *Green Computing in Smart Cities: Simulation and Techniques. Green Energy and Technology.*, B. Balusamy, N. Chilamkurti, S. Kadry. Eds. Manhattan, NYC, USA: Springer, Cham, 2021, pp. 35–50, doi: 10.1007/978-3-030-48141-4_3.

[8] Y. P. Huang and H. Basanta, "Bird image retrieval and recognition using a deep learning platform," *IEEE Access*, vol. 7, pp. 66980–66989, 2019, doi: 10.1109/ACCESS.2019.2918274.

[9] Y. LeCun, L. eon Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proc. IEEE*, vol. 86, no. 11, pp. 2278-2324, 1998.

[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv e-print* pp. 1409.1556, 2014. [Online], Available: https://ui.adsabs.harvard.edu/abs/2014arXiv1409.1556S.

[11] A. Satyo, B. Karno, D. Arif, and I. Sari, "Deteksi covid-19 image chest x-ray dengan convolution neural network efficient net-b7," (in Indonesia), in *Semin. Nas. Teknol. Inf. dan Komun. STI&K*, vol. 5, no. 1, pp. 2581–2327, 2021.

[12] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2: inverted residuals and linear bottlenecks," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 4510–4520, 2018, doi: 10.1109/CVPR.2018.00474.

[13] A. G. Howard *et al.*, "MobileNets: efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. 1704.04861, 2017, [Online]. Available: http://arxiv.org/abs/1704.04861

[14] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, pp. 8697–8710, 2018, doi: 10.1109/CVPR.2018.00907.

[15] M. Tan and Q. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," in *Proc. 36th Int. Conf. Mach. Learn.*, Long Beach, California, USA, 2019, vol. 97, pp. 6105–6114. [Online]. Available: https://proceedings.mlr.press/v97/tan19a.html

[16] M. Tan and Q. V. Le, "EfficientNetV2: smaller models and faster training," in *Proc. 38th Int. Conf. Mach. Learn*, 2021, [Online]. Available: http://arxiv.org/abs/2104.00298.