

Image preprocessing and hyperparameter optimization on pretrained model MobileNetV2 in white blood cell image classification

Parmonangan R. Togatorop¹, Yohanssen Pratama², Astri Monica Sianturi¹, Mega Sari Pasaribu¹,
Pebri Sangmajadi Sinaga¹

¹Department of Information System, Faculty of Informatics and Electrical Engineering, Institut Teknologi Del, Laguboti, Indonesia

²Department of Software Engineering Technology, Faculty of Vocational Studies, Institut Teknologi Del, Laguboti, Indonesia

Article Info

Article history:

Received Sep 1, 2021

Revised Sep 10, 2022

Accepted Sep 21, 2022

Keywords:

Convolutional neural network

Hyperparameter optimization

Image classification

Image preprocessing

MobileNetV2

Transfer learning

White blood cell

ABSTRACT

White blood cells play a role in maintaining the immune system which consists of several types such as neutrophils, lymphocytes, monocytes, eosinophils and basophils. MobileNetV2 is one of the pretrained convolutional neural network (CNN) models that provides excellent advantages and performance in classifying images. In this research was conducted to find out how to apply optimization hyperparameters and the impact of image processing on white blood cell image classification using MobileNetV2, so that it is expected to find a combination of preprocessing and combination of hyperparameter values that can produce the highest accuracy value. To maximize the classification process, before classifying the image, several stages of image preprocessing are carried out, namely cropping, grayscale, resizing and augmentation. Hyperparameter tuning was carried out for an experiment to improve model performance. The three main parameters used in hyperparameter tuning are learning rate, batch size, and number of epochs. Performance optimization model performance will be measured using accuracy, sensitivity, specificity and using a confusion matrix. Based on the experimental results in this study, it shows that the best learning rate value is 0.00001, the best batch size value is 32, and the best epoch value is 250.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Yohanssen Pratama

Department of Software Engineering Technology, Faculty of Vocational Studies,

Institut Teknologi Del

Jl. Sisingamangaraja Sitoluama-Laguboti, Toba 22381, Indonesia

Email: yohanssen.pratama@del.ac.id

1. INTRODUCTION

White blood cells play a role in maintaining human immunity. The types of white blood cells are neutrophils, lymphocytes, monocytes, and eosinophils. Currently, there have been many studies related to white blood cell image classification [1]. Each type of white blood cell image has a different shape and morphology from texture, shape and size. [2]. Yildirim and Çinar [3] classified white blood cell images into 4 types, namely eosinophils, lymphocytes, monocytes, and neutrophils using Alexnet, Resnet50, Densenet201 and GoogleNet. The results of the classification used AlexNet 79.27%, Resnet50 78.74%, DenseNet201 77.88%, and GoogleNet 62.93% [3], [4] classified white blood cell images into 4 types (lymphocytes, monocytes, eosinophils, and neutrophils) using convolutional neural network (CNN), LeNet and AlexNet. The classification accuracy results obtained are CNN 75.36%, LeNet 65.21% and AlexNet 80.37% [4].

Throngnumchai *et al.* [2] classified white blood cell images into 4 types, namely Monocytes, Neutrophils, Eosinophils, and Lymphocytes using MobileNetV2 with an accuracy of 93.18% [2]. Based on several previous studies, it was shown that the pretrained model of MobileNetV2 has good performance for classifying white blood cell images.

MobileNetV2 can be considered as an appropriate technique to overcome the problem of image classification [5]. Dong *et al.* [5] compared the MobileNetV1, MobileNetV2, and CNN models. Compared to the three models used, MobileNetV2 has the highest accuracy value of 89.00%. Tobias *et al.* [6] concluded that the application of CNN, especially the pretrained model of MobileNetV2 with Linear Bottleneck, can achieve 90% accuracy [6].

MobileNetV2 has a number of hyperparameters that can produce different classification accuracy for each of the same tasks with different hyperparameters. Yuningsih and Mustikasari [7] classified anemia based on red blood cell morphology using the CNN. In this study, experiments were conducted on hyperparameters. The hyperparameters used for the training algorithm are epoch, mini-batch, learning rate. with an accuracy of 0.9774 [7], [8] conducted a study on the effect of batch size on the performance of CNNs for various datasets. Xu [9] using hyperparameter learning rate, batch size, and number of epochs. The experimental results show that when the other hyperparameters remain unchanged, the error trained by the network model decreases significantly as the learning rate increases, the batch size decreases and the number of epochs accumulates within a certain range [9], [10] conducted research on the importance of using hyperparameter tuning. Machine learning algorithms such as neural networks involve a number of hyperparameters that must be defined before they are executed. Hyperparameters can be determined during training and need to be optimized to achieve maximum performance. Determination of hyperparameters can minimize generalization errors. Therefore, the results of image classification accuracy using CNN architecture with pretrained MobileNetV2 model can be optimized by optimizing hyperparameters.

To support the model's performance in classifying images, there are several factors that influence in terms of data availability, namely the image focused on the object [11], the amount of data processed is sufficient [12], the image can be processed with small computations [11], and so on. In a study using medical images, [11] detected a disease of narrowing of blood vessels in the brain. This study uses several pre-processing techniques, namely image scaling, image gray-scaling, and image noise removal using several techniques. In medical image processing, image pre-processing is very important so that the extracted image does not have noise [2], [11] apply image augmentation and morphology image processing before classifying. Efficient classification of white blood cell types requires a lot of information for study. This researcher reproduced the data up to 10,000 images. Novoselnik *et al.* [12] classifying white blood cell images using the CNN model where one of the preprocessing techniques used is data augmentation. The available dataset contains 738 images of blood samples, the dataset is very unbalanced (50% of the dataset is neutrophils) which can result in a biased model. Therefore, augmentation techniques are used, namely random rotation, horizontal shift, vertical shift, and horizontal flipping [12]. That results in a higher variance of the training data, which prevents the model from overfitting. Throngnumchai *et al.* [2] perform morphology image preprocessing and data augmentation in white blood cell image classification to improve the quality and increase the number of white blood cell images. The white blood cell image which initially numbered 410 was augmented to 10,000 data [2]. Based on the literature study that has been described, the researcher will perform several image preprocessing techniques and hyperparameter optimization to improve the performance of the pretrained MobileNetV2 model in classifying white blood cell images.

2. RESEARCH METHOD

This section describes the research methods used to classify white blood cell images. The first step is to download the data to be used from the Kaggle website, that is the white blood cell image dataset. Then perform several stages of image processing to improve image quality. Transfer learning from pre-trained MobileNetV2 was carried out and a hyperparameter optimization algorithm was applied to classify white blood cell image types. Then evaluate the results of the classification of the type of white blood cell image obtained previously.

The research was conducted using the MobileNetV2 Pretrained Model. MobileNetV2 is a CNN architecture that works based on an inverted residual structure, where the input and output of the residual block are bottleneck layers as opposed to the traditional residual model, which uses an extended representation in the input. The MobileNetV2 architecture consists of 32 fully convolution layers with 32 filters, followed by 19 residual bottleneck layers. The MobileNetV2 model runs faster for the same accuracy across the latency spectrum. Specifically, the new model uses twice less operation, requires 30% fewer parameters and is about 30-40% faster on Google Pixel phones than the MobileNetV1 model [13]. In MobileNet there is a special layer known as depthwise separable convolution and could be seen in Figure 1. This layer plays a role in reducing

complexity and parameters so that it will produce a model with a larger size. In the last stage we will apply pointwise convolution to reduce the dimensions, the pointwise convolution could be seen in Figure 2.

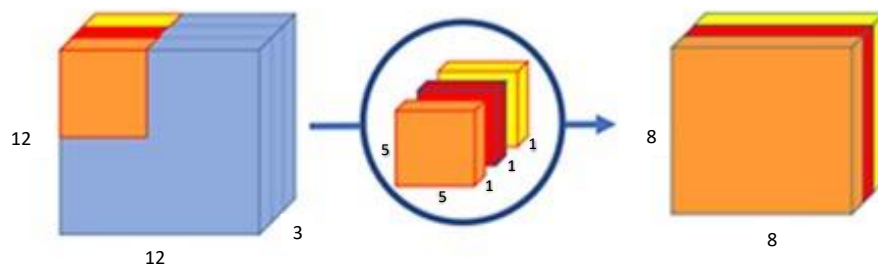


Figure 1. Depthwise convolution

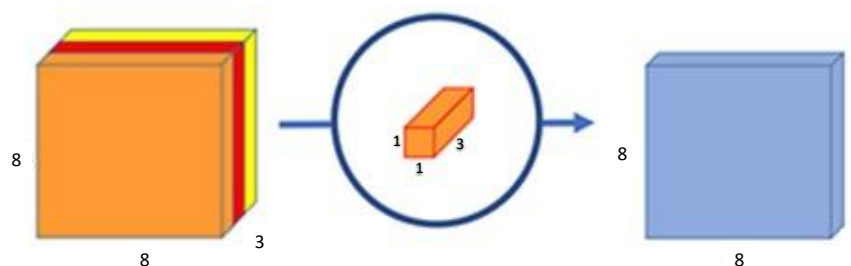


Figure 2. Pointwise convolution

One of the developments from the previous version of MobileNet is the Linear Bottleneck. The linear bottleneck function is to remove the nonlinear activation function from the 1×1 pointwise convolution, to maintain the features that have been extracted by the convolution, the linear activation function will be used. The use of linear bottlenecks aims to ensure the expressiveness of the model. To reduce the loss of information, the output of the dimension layer reduction, namely the bottleneck layer is not connected to the non-linear activation layer, but is connected to the Linear Bottleneck [14], [15].

MobileNetV2 has been used by several researchers to classify images and the classification results obtained are good. MobileNetV2 which uses a linear bottleneck can achieve higher accuracy with much less processing time making it more efficient. Some of the reasons that support researchers using MobileNetV2 are considered from several advantages [16], [17].

- MobileNetV2 is able to achieve higher accuracy with much less processing time so that it is more efficient than the CNN model.
- MobileNetV2 has a smaller number of parameters than the original MobileNet so less computational costs are required.
- MobileNetV2 is a very effective feature extractor for image classification, object detection, and segmentation.
- MobileNetV2 is an architecture that optimizes memory consumption.

The following Figure 3 is the MobileNetV2 architecture which is composed of several layers. Based on Figure 3, it can be seen that MobileNetV2 is composed of input layer, functional layer, average pooling, and dense layer. The input layer will input $224 \times 224 \times 3$. The output of the input layer is an array with a size of $224 \times 224 \times 3$. The output of the input layer will be processed into the functional layer. The functional layer consists of several bottlenecks. The number of bottlenecks in MobileNetV2 is 17 blocks. Each bottleneck will receive and process a different input size. The output of the functional layer is an array with a size of $7 \times 7 \times 1280$. Then the results of the Functional layer will be processed on the next layer, namely GlobalAveragePooling2D. In GlobalAveragePooling2D will be given a 7×7 filter. The channel given to this layer is None so that the output size is $1 \times 1 \times 1280$. In the Dense Layer, the four types of white blood cells will be classified. Every bottleneck in MobileNetV2 has its constituent components. The following Table 1 are the components that make up the bottleneck in MobileNetV2. The research method used to classify white blood cell images can be seen in the following Figure 4.

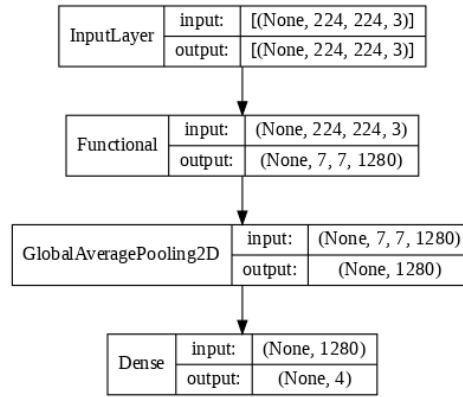


Figure 3. The Architecture of MobileNetV2

Table 1. The components of bottleneck in MobileNetV2

No	Bottleneck	Components
1	Bottleneck I	Depthwise Conv2D Batch Normalization ReLU Conv2D
2	Bottleneck II, IV, VII, XIV	Batch Normalization Conv2D Batch Normalization ReLU Zero Padding Depthwise Conv2D Batch Normalization ReLU Conv2D
3	Bottleneck III, V, VI, VIII, IX, X, XI, XII, XIII, XV, XVI, XVII	Batch Normalization Conv2D Batch Normalization ReLU Depthwise Conv2D Batch Normalization ReLU Conv2D Batch Normalization

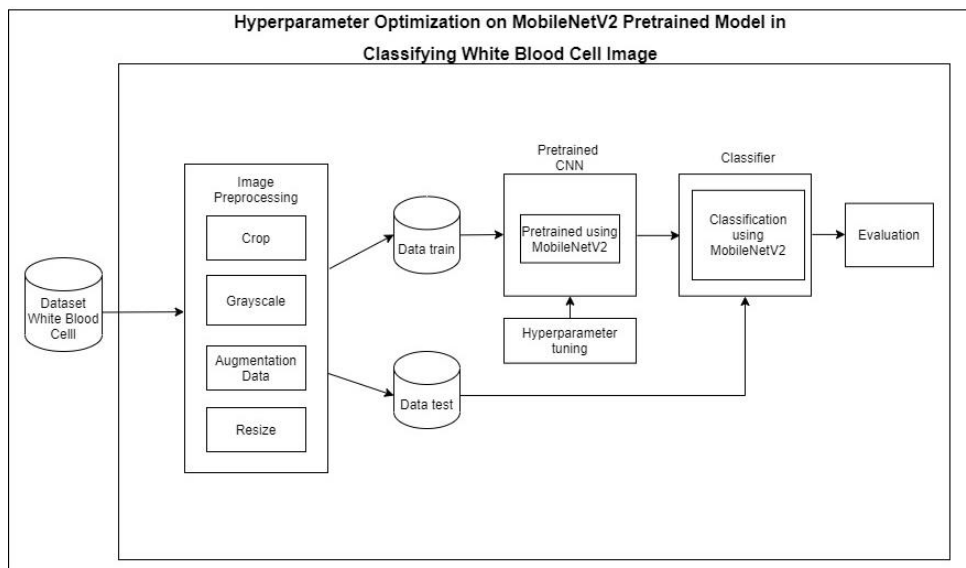
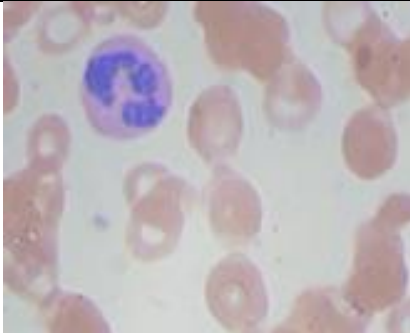
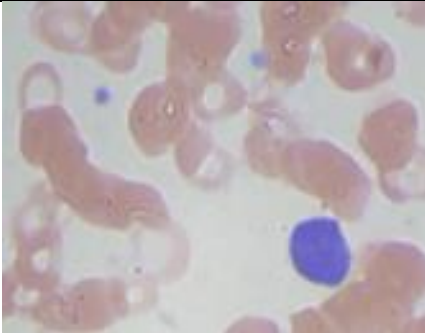
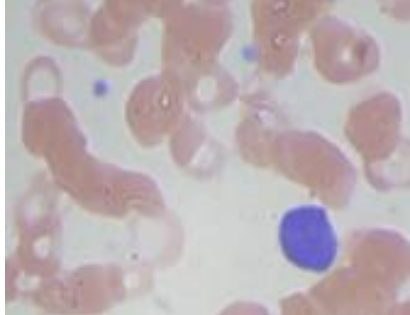
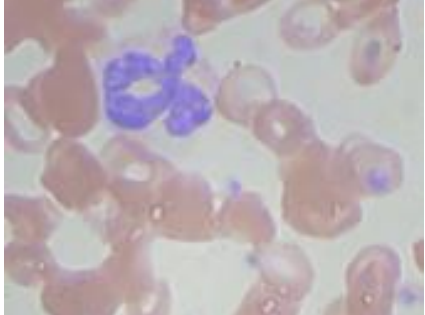


Figure 4. The general stages of the system in the research

2.1. Provide white blood cell image dataset

To train and evaluate MobileNetV2, we have utilized the dataset available from the Kaggle website. The dataset stores 367 images consisting of five types of white blood cells, namely basophils, eosinophils, lymphocytes, monocytes, and neutrophils. However, in this study, 4 types of white blood cells were used, namely eosinophils, lymphocytes, monocytes and neutrophils. Available images have been labeled; each picture contains 1-2 pictures of white blood cells. The white blood cell object in the image file is located in a non-uniform position. The white blood cell images in the dataset are composed of Red, Green, Blue (RGB) color modes. The following Table 2 is an image of a white blood cell obtained from Kaggle that will be used in this study.

Table 2. Example of white blood cell image on the dataset

Image	Cell Type	Image	Cell Type
	Eosinophils		Monocytes
	Lymphocytes		Neutrophils

2.2. Image preprocessing

Image pre-processing plays an important role in supporting image processing. This is necessary to simplify processing because most of the image data is impractical to manipulate. With the implementation of image preprocessing some irrelevant image features will be reduced to make processing and analysis tasks more effective and efficient. There are several stages applied in the image preprocessing stage in this study, namely resizing, cropping, grayscaling, and data augmentation.

2.2.1. Cropping

Cropping is an image cutting process that is used to remove other components that are not included in the object of study. Cropping is done to focus the main image that will be used in image processing. Before the image is cropped, the image position (right, left, up, down) will be adjusted according to the desired pixels to get an image that focuses on the desired main object. After being analyzed there are several images that contain many other components besides the white blood cell object. The purpose of cropping is to focus on the main object that you want to extract and eliminate components or noise that are not included in the focus of the research object [18], [19].

Cropping will be done by using the Numpy and cv2 libraries. At the cropping stage, the bounding box will be used by utilizing the region of interest (RoI) function. First, the image will be loaded, then define the RoI function and then determine the bounding box of the image to be cropped. Cropping will be carried out on all 367 raw white blood cell images by utilizing the RoI function, 367 new white blood cell images will be generated as well. This cropping process will be repeated 367 times according to the amount of raw data you want to crop using the same configuration. The following Figure 5 is the result of the cropping process.

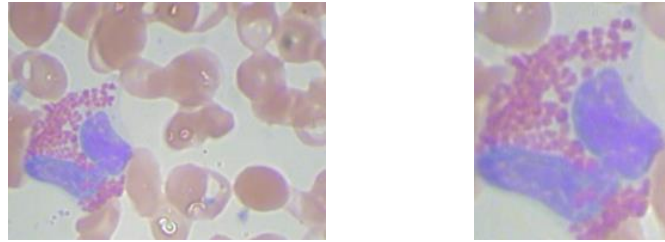


Figure 5. The results of cropping technique on image of white blood cells

2.2.2. Grayscale

Grayscale is applied to convert the RGB image color to grayscale. Grayscale helps to simplify the RGB layer of the image containing the image information at each layer and obtain the image in grayscale intensity levels. The application of grayscale on white blood cell images will make computational time efficient, the storage space required when running the algorithm [11]. The color conversion that will be carried out in this study will use the formula [20],

$$I = 0.299 R + 0.587 G + 0.114 B$$

the color conversion to grayscale will be done by using the cv2 libraries. This grayscale process will be repeated as many as 367 according to the amount of raw data that you want to grayscale using the same configuration. The image conversion process begins by loading the previously cropped image as grayscale input. Then convert the image color using a formula. The results of the grayscale image will then be saved. The data used as input in the grayscale process is the image of white blood cells resulting from the cropping process that has been done previously. The following Figure 6 is the result of the grayscale process.

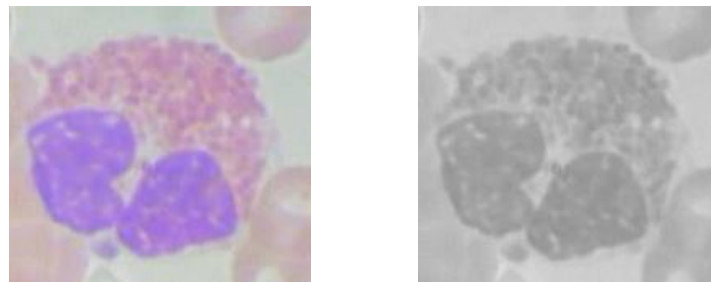


Figure 6. The results of grayscale technique on image of white blood cells

2.2.3. Data augmentation

The data available on the sources used still amount to 367 files. This amount of data is very less to be used in the process of implementing image classification so that the augmentation process is needed to produce more data to get more optimal performance [21]. After the preprocessing of cropping and grayscale is done, the next step will be the augmentation stage. The augmentation stage carried out in this study will produce 8,000 train data images, each type 2,000, and test data will be generated as many as 2,000 images, each type 500.

Image augmentation in python can be done using the ImageDataGenerator function provided by Keras. The techniques used for this augmentation are random horizontal flip, random vertical flip, random image rotation, and zooming. The data used as input in the augmentation process is the image of white blood cells resulting from the grayscale process that has been done previously. The data will be loaded by defining the path of the image directory location and then checking the image size. The size of the image is a 3-dimensional array (224, 224, 3) so it is necessary to change the size of the image dimensions to a 4-dimensional array by adding 1 dimension, this is because the NumpyArrayIterator must have a 4-dimensional array so that the result of the image array size becomes (1, 224, 224, 3). The next step is to produce n images. The following Figure 7 is the result of the image augmentation process where 1 image is augmented into 5 images.

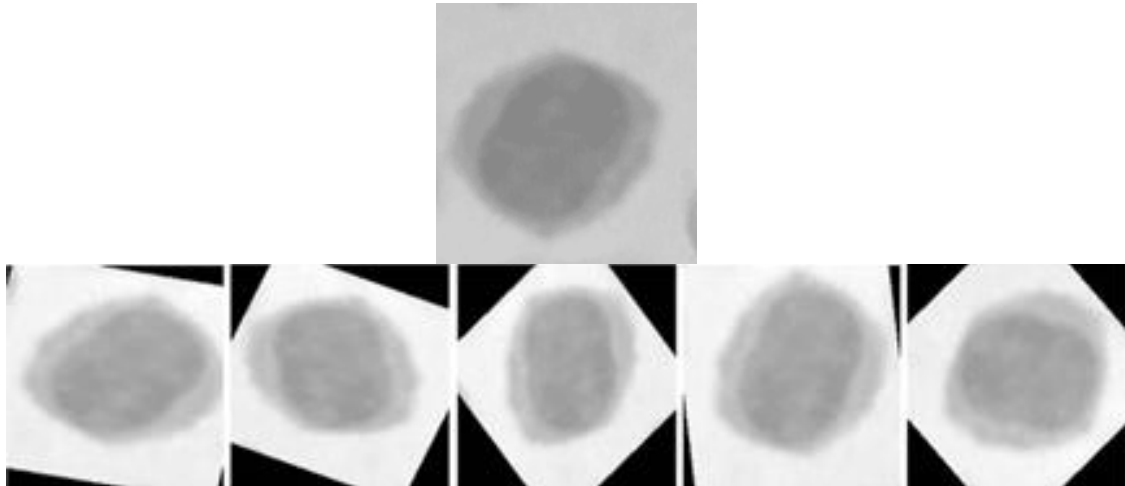


Figure 7. The results of augmentation technique on image of white blood cells

2.2.4. Resizing

Resizing is used to equalize the size of all images that will be used in image processing. Resizing the image can speed up the computational process and increase the effectiveness of model training as the image size gets smaller [18]. The application of image resizing makes the results of feature extraction consistent. In MobileNetV2, the image size usable on the model is larger than 32×32 . However, the larger the image size, the better the model performance will be. The image size that will be used for this research is 224×224 . The data used as input in the resizing process is the image of white blood cells resulting from the augmentation process that has been done previously.

Image resizing can be done by using the cv2 and Glob libraries. The program will read the image of white blood cells resulting from the augmentation process and resize the image according to the input dimensions using the cv2.resize function. In this study, the image size that will be used is 224×224 . The following Figure 8 is the result of the image resizing.

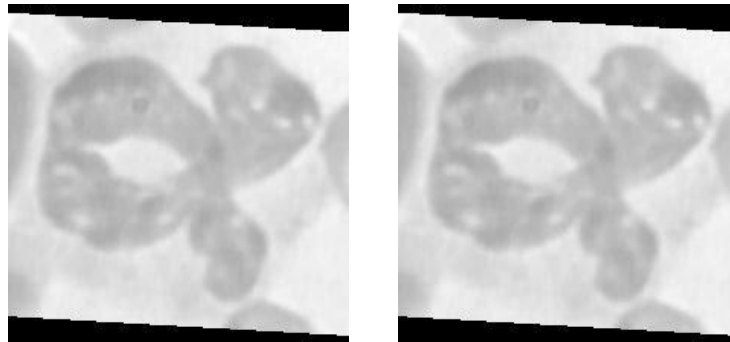


Figure 8. The results of resize technique on image of white blood cells

2.3. Transfer learning from pretrained model

The use of transfer learning is aimed at improving the performance of the model that will be used in white blood cell image classification. The use of transfer learning makes it easier for researchers not to carry out the training process from the beginning. In this study, researchers can use the results of the pre-trained model to extract features from all white blood cell images in a predetermined dataset [5], [6].

In this study, transfer learning from the Pretrained Model MobileNetV2 will be used to classify white blood cell images. Researchers will implement the system by loading MobileNetV2 using Keras tools and the Tensorflow backend on the initial layer, then adding a dense layer to classify white blood cell types. At each layer on MobileNetV2, the research team used Imagenet as the weight. All weights in the pre-trained model (MobileNetV2) have been trained with ImageNet to recognize colors, textures, and other patterns. In addition

to using the pretrained CNN model as a feature extractor and retraining the classifier layer that is fully connected. Transfer learning can also be done by fine-tuning the weights of the pretrained model [22].

In this study, researchers will do fine-tuning to retrain several convolution layers that have been determined in number. So that the model will obtain a feature extraction result that is more in line with the data set because it is obtained from the results of retraining several convolution layers on fine-tuning. The application of fine-tuning can improve the performance of the model in image classification. In fine-tuning the model will extract and re-learn the features of the image on the tuned layer to extract specific features that are more suitable for the dataset so as to improve the model's performance in classifying white blood cell images according to each type.

2.4. Applying hyperparameter optimization

At this stage the hyperparameter optimization is carried out using the manual hyperparameter tuning method where the hyperparameter values used will be set manually by combining hyperparameters [23]. The hyperparameters used in this research are learning rate, batch size and epoch [24]. This research will be conducted experimentally. Each experiment will collect the results and rank based on the value of accuracy and loss.

The experiments were carried out sequentially, starting with the learning rate experiment. After obtaining the best learning rate value, then the batch size experiment was carried out. After obtaining the best batch size value, the epoch experiment was carried out. The experimental learning rate values are 0.1, 0.01, 0.001, 0.0001, 0.00001, 0.000001, 0.0000001. The values of the batch size that were experimented with were 32, 64, 128, and 256. The epoch values that were experimented with were 50, 100, 150, 200, 250.

2.5. Evaluation of experimental results

After the experiment is carried out, then an evaluation of the model that has been built is then carried out. In the evaluation process starting from, the data to be processed is the white blood cell dataset. Data first by dividing the data into train data and test data. Each training and testing process will be evaluated to see how well the built model classifies the white blood cell image. Evaluation of training and testing is done by calculating the accuracy value of each model. The test data will be evaluated using the library confusion matrix to calculate the values of accuracy, sensitivity and specificity [25].

3. RESULTS AND DISCUSSIONS

The results and discussions sections explain the results of the implementation and discuss the results obtained from the implementation system. In this part, the implementation of the design that have been made will be explained. The things that will be explained in this part are the implementation environment, implementation constraints, libraries, image preprocessing implementation, CNN model implementation and model evaluation and how the results obtained from each classification process carried out in this study.

3.1. Preprocessing result

We have tested the effect of preprocessing on the resulting accuracy value. The test scenario carried out is to first test the effect of preprocessing on the accuracy value if the preprocessing technique is only resizing and augmentation. After that, testing the effect of preprocessing on the accuracy value if the preprocessing technique used is cropping, resizing and augmentation. Based on the Table 3, it can be seen that the combination of preprocessing cropping, grayscale, augmentation and resizing has a more optimal accuracy value by considering the resulting test value.

Table 3. Average accuracy results of the preprocessing experiment

Experiment	Train		Validation		Test	
	Accuracy	Loss	Accuracy	Loss	Accuracy	Loss
Resize + Augmentation	0.8670	0.3865	0.8302	0.4566	0.4770	1.6311
Cropping + Resize + Augmentation	0.9947	0.0243	0.9906	0.0325	0.8470	0.8803
Grayscale + Resize + Augmentation	0.8662	0.3886	0.8315	0.4517	0.5360	1.5360
Cropping + Grayscale + Resize + Augmentation	0.95774	0.13882	0.93633	0.17748	0.9665	0.0907

In the resize and augmentation experiments, the accuracy value obtained is quite good, but in the test results, the accuracy value is very low. In the cropping, resizing and augmentation experiments, the accuracy value obtained is good, but in the test results, the accuracy value is quite good. In the grayscale, resize and augmentation experiments, the accuracy value obtained is quite good but in the test results, the accuracy value

is very low. In the cropping, grayscale, resize and augmentation experiments, the accuracy values obtained are good and in the test results, the accuracy values are also good.

3.2. Hyperparameter optimization

The selection of the appropriate hyperparameter is one of the things that plays a very important role in CNN training and has a major impact on the performance of the model being built. We conducted several experiments to select the best hyperparameter values in classifying white blood cell images. The following are the results and discussion of hyperparameter optimization experiments on learning rate, batch size, and number of epochs.

3.2.1. Learning rate

Experiments on learning rate are used to determine the learning rate that produces the highest best accuracy value. The experimental learning rate value is 0.1, 0.01, 0.001, 0.0001, 0.00001, 0.000001, 0.0000001. Based on the Table 4, it shows the learning rate value affects the model's performance in white blood classification.

Table 4. Average validation accuracy results of the learning rate experiment

Experiment	Learning Rate	Batch Size	Number of Epochs	Average Value of Validation Accuracy	Average Value of Validation Loss
1	0.1	32	60	0.69611	3.26628
2	0.01	32	60	0.91362	1.79781
3	0.001	32	60	0.97009	0.41778
4	0.0001	32	60	0.99037	0.03675
5	0.00001	32	60	0.99098	0.02607
6	0.000001	32	60	0.95049	0.15984
7	0.0000001	32	60	0.53752	1.07380

From the experiments, the learning rate with the highest accuracy was obtained at the application of learning rate 0.00001 (1×10^{-5}) with an accuracy 0.99098 and a loss value 0.02607. The following Figure 9 is a graph of the average accuracy and loss validation values for the learning rate experiments. Based on the graph, it can be seen that in the learning rate range of 0.1 - 0.00001, the accuracy value tends to be higher and the loss value is getting smaller. This is due to a decrease in the value of the learning rate from 0.1 to 0.00001 which results in a better model recognizing data, so that the resulting accuracy increases and the loss decreases.

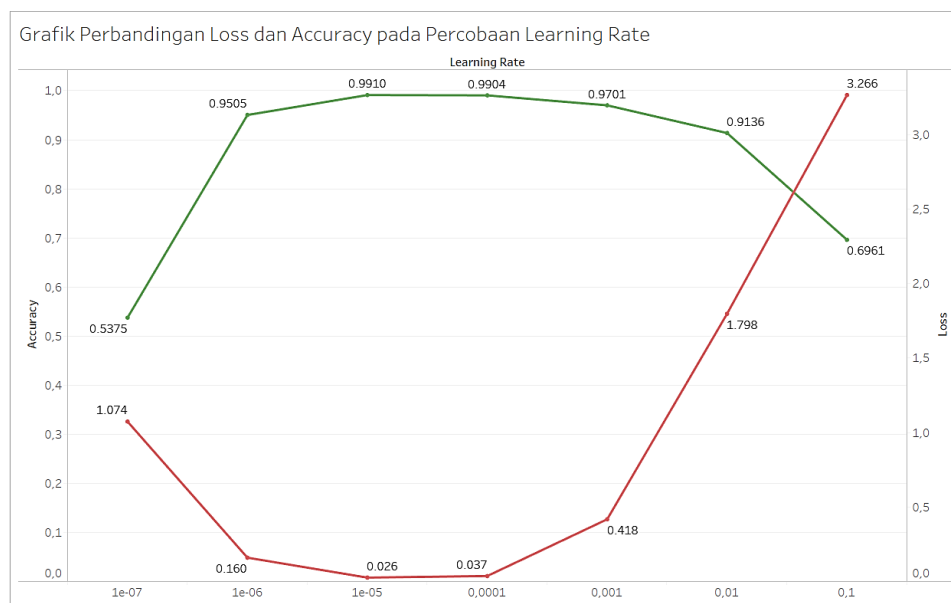


Figure 9. Graph of average validation accuracy and loss from the learning rate experiment

The model training uses a learning rate of 0.001; 0.0001, 0.00001 produces high accuracy and small loss. This is because the use of a small learning rate value makes the model speed better at recognizing data. However, the use of a small learning rate will require a lot of time in conducting training. Meanwhile, model training using learning rates of 0.1 and 0.01 resulted in poor accuracy and significant loss. This is because the use of a large learning rate value makes the model speed faster in recognizing data. The use of a large learning rate tends to require a short time in conducting training. However, based on experiments on the learning rate, after training the model uses a smaller learning rate value of 0.000001 and 0.0000001, the accuracy value decreases (smaller) and the loss value increases. Therefore, it can be seen that the model has a good performance in conducting training not depending on the size of the learning rate but when the learning rate is at the optimal level.

3.2.2. Batch size

In the batch size experiment, 60 epochs and a learning rate of 0.00001 were used, which is the learning rate value with the best accuracy in the previous learning rate experiment. The following are the results and discussion of the batch size experiment carried out for the values of 32, 64, 128 and 256. Based on the Table 5, it is known that the batch size value affects the model's performance in classifying white blood cell images.

Table 5. Average validation accuracy results of the batch size experiment

Experiment	Batch Size	Learning rate	Number of Epochs	Average Value of Validation Accuracy	Average Value of Validation Loss
1	32	0.00001	60	0.99097	0.02607
2	64	0.00001	60	0.98251	0.05420
3	128	0.00001	60	0.96100	0.13669
4	256	0.00001	60	0.90955	0.31664

The smaller the batch size, the more optimal the accuracy value and the smaller loss value. From the experiments that have been carried out, it is known that the most optimal batch size=32 value is with an accuracy value of 0.99097 and a loss value of 0.02607. The following Figure 10 is a graph of the average accuracy and loss validation values for the batch size experiments that have been carried out. Based on the experimental batch size chart in Figure 10, it can be seen that the smaller the batch size, the higher the accuracy value and the smaller the loss value and vice versa.

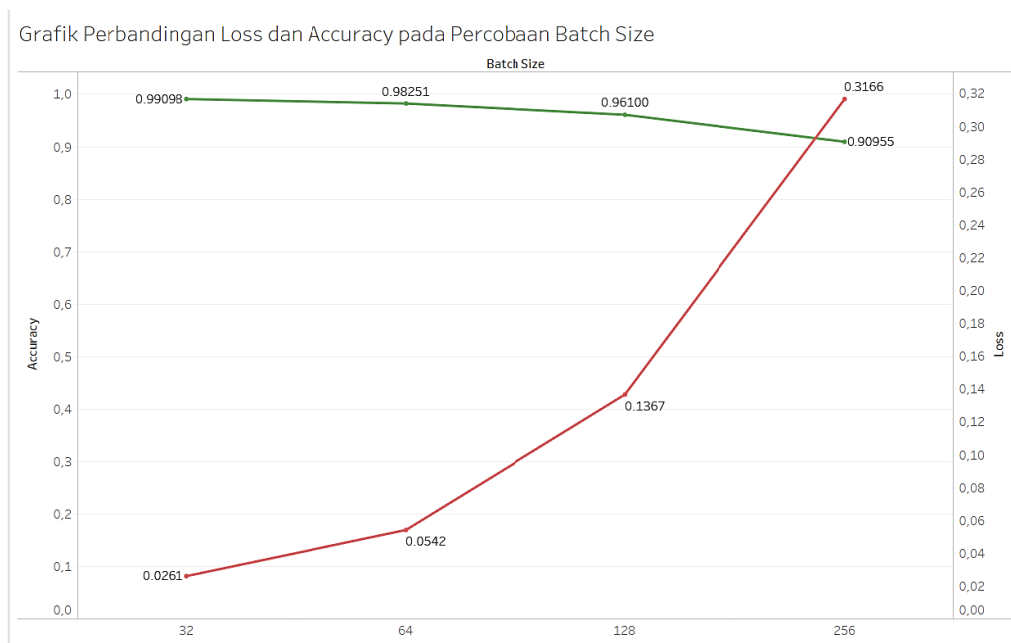


Figure 10. Graph of average validation accuracy and loss from the batch size experiment

This is because the batch size experiment was carried out using a small learning rate of 0.00001. The use of a learning rate value of 0.00001 which is included in the small category, will require a small batch size value as well in order to maintain the stability of the model. Therefore, the value of batch size 32 will be more optimal than other batch sizes for the use of a learning rate of 0.00001 which is included in the small category. Theoretically, batch size affects training time but does not significantly affect test performance, therefore the accuracy value in each experiment does not change significantly.

3.2.3. Number of epoch

To determine the number of epochs that produce the highest best accuracy value, the researcher conducted experiments on several epochs, namely 50, 100, 150, 200, 250. In this experiment, the learning rate and batch size used were selected based on the learning rate experiment and batch experiment. size with a learning rate of 0.00001 and a batch size of 32. Based on the experiments that have been carried out, the researchers obtained the average value of the validation accuracy from the number of epoch experiments which can be seen in the following Table 6.

Table 6. Results of the number of epoch experiment

Experiment	Epoch	Learning Rate	Batch size	Average Value of Validation Accuracy	Average Value of Validation Loss
1	50	0.00001	32	0.98719	0.01547
2	100	0.00001	32	0.99525	0.00844
3	150	0.00001	32	0.99749	0.00695
4	200	0.00001	32	0.99786	0.00488
5	250	0.00001	32	0.99892	0.00380

After conducting several experiments on the number of epochs, namely 50; 100; 150; 200; 250, the researcher obtained the highest average validation accuracy value in the experiment with a number of epochs of 250. From the graph in Figure 11, it can be seen that the larger the epoch, the higher the accuracy value and the smaller the loss value and vice versa. This statement is proven in the experimental results, namely at epoch 50, the accuracy value is 0.98719 and the loss is 0.01547. As the epoch increases, the accuracy increases and the loss decreases. So that the epoch 250 obtained an accuracy value of 0.99892 and a loss value of 0.00380. From the experimental results, it can be seen that the number of epochs does not have a significant effect on the accuracy and loss values. This is because the epoch hyperparameter represents the number of iterations the learning algorithm will work through the entire training dataset. Therefore, the more iterations given, the better the accuracy value.

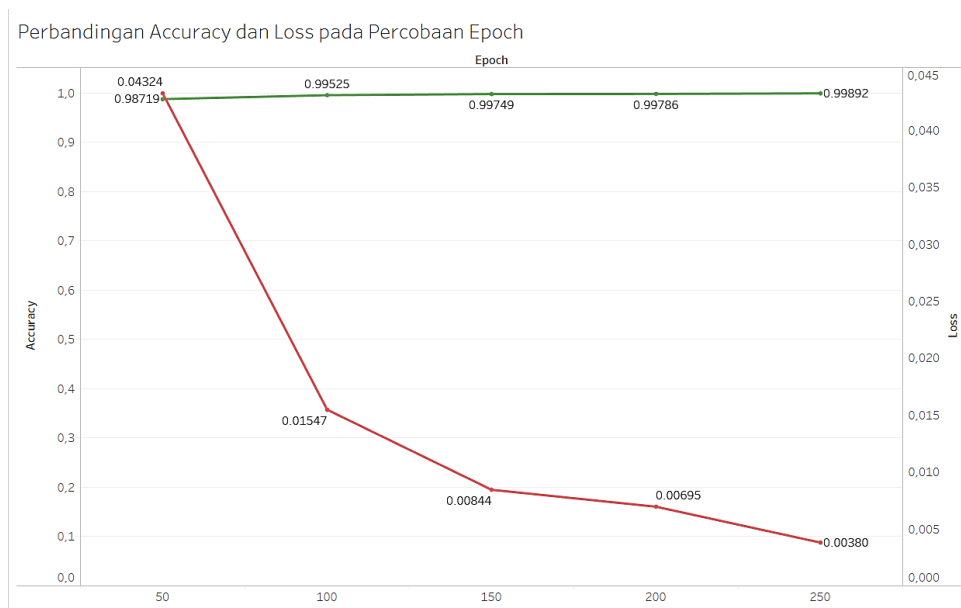


Figure 11. Graph of average validation accuracy and loss from the number of epoch experiment

4. EXPERIMENTAL EVALUATION

After the experiment is done, then testing is carried out on the model that has the best accuracy value. The test results will be evaluated using a confusion matrix. The test data used consisted of 2,000 image data, namely 500 images of each type of white blood cell tested.

The evaluation results obtained in each learning rate experiment using a batch size of 32, number of epochs 60 and a learning rate value of 0.01-0.000001 resulted in a relatively good test evaluation (testing) indicated by a good accuracy value, which is close to 1. In this study, the best average model is obtained from the highest average value in all iterations of the fold. Test results to ensure that the model used is good for recognizing images with different data sets.

The results of the evaluation carried out on each batch size experiment, it can be seen that for the learning level of 0.00001 and epoch 60, the batch size of 32-256 produces a relatively good accuracy value on the evaluation of the test (test). In this study, determining the best model is done by finding the average value of accuracy in all fold iterations. The test results are used to ensure that the model used performs well to recognize images with different data sets.

From the epoch experiment, it is known that for a learning rate of 0.00001, batch size of 64, and epochs 60-250, the evaluation of testing is relatively good which shows the accuracy is close to 1. by finding the average value across the fold iterations. The test results are used to ensure that the model used performs well to recognize images with different data sets.

5. CONCLUSION

MobileNetV2 can be optimized by using hyperparameter optimization manually to classify white blood cell images by changing each hyperparameter value for each experiment performed. Based on the experimental results, it shows that the best learning rate value is 0.00001, the best batch size value is 32, and the best epoch value is 250. The use of a relatively small learning rate results in slower model speed in learning, but a smaller learning rate allows the model to recognize the data better. The use of a smaller learning rate will require a longer learning time. When the learning rate is higher, the speed of the model increases when it recognizes data because a large learning rate requires a faster time to train. The smaller the batch size value, the higher the accuracy value. A high accuracy value will be directly proportional to the smaller loss value and vice versa. Batch size does not really show a significant effect on test performance when choosing the right learning rate. The use of a small learning rate will require a small batch size to produce maximum accuracy. The greater the epoch value, the higher the accuracy value generated, the greater the accuracy value will result in a smaller loss value and vice versa. Based on the experiments that have been carried out, the researcher concludes that the improvement in model performance is not only influenced by hyperparameter optimization but also by image preprocessing. Hyperparameter optimization is able to improve model performance, but the accuracy value obtained is less than optimal. Therefore, researchers perform image preprocessing to improve the results of model performance. The data used in this study has been processed in such a way as described in the image preprocessing design.

ACKNOWLEDGEMENTS

This work was supported and funding by the research institutions and community service of Institut Teknologi Del.




REFERENCES

- [1] A. D. Ware, "The complete blood count and white blood cell differential," *Contemporary Practice in Clinical Chemistry*, pp. 429–444, 2020, doi: 10.1016/b978-0-12-815499-1.00025-9.
- [2] K. Throngnumchai, P. Lomvisai, C. Tantasirin, and P. Phasukkit, "Classification of white blood cell using deep convolutional neural network," *BMEiCON 2019-12th Biomedical Engineering International Conference*, 2019, doi: 10.1109/BMEiCON47515.2019.8990301.
- [3] M. Yildirim and A. Çinar, "Classification of white blood cells by deep learning methods for diagnosing disease," *Revue d'Intelligence Artificielle*, vol. 33, no. 5, pp. 335–340, 2019, doi: 10.18280/ria.330502.
- [4] K. Kousalya, S. Santhiya, K. Dinesh, R. S. Mohana, and B. Krishnakumar, "Blood cells classification using convolutional neural network architecture," *International Journal of Advanced Science and Technology*, vol. 29, no. 3 Special Issue, pp. 261–267, 2020.
- [5] K. Dong, C. Zhou, Y. Ruan, and Y. Li, "MobileNetV2 model for image classification," *Proceedings-2020 2nd International Conference on Information Technology and Computer Application, ITCA 2020*, pp. 476–480, 2020, doi: 10.1109/ITCA52113.2020.00106.
- [6] R. R. N. M. I. Tobias et al., "CNN-based deep learning model for chest X-ray health classification using TensorFlow," *Proceedings-2020 RIVF International Conference on Computing and Communication Technologies, RIVF 2020*, 2020, doi: 10.1109/RIVF48685.2020.9140733.
- [7] N. Yuningsih and M. Mustikasari, "Anemia classification based on abnormal red blood cell morphology using convolutional neural network," *IOSR J. Comput. Eng.*, vol. 22, no. 1, pp. 22–29, 2020, doi: 10.9790/0661-2201042229.




- [8] P. M. Radiuk, "Impact of training set batch size on the performance of convolutional neural networks for diverse datasets," *Information Technology and Management Science*, vol. 20, no. 1, 2018, doi: 10.1515/itms-2017-0003.
- [9] N. Xu, "Research on the influence of convolutional neural network parameters on fruit classification," *2019 IEEE 3rd International Conference on Electronic Information Technology and Computer Engineering, EITCE 2019*, pp. 606–610, 2019, doi: 10.1109/EITCE47263.2019.9094902.
- [10] P. Probst, A. L. Boulesteix, and B. Bischl, "Tunability: Importance of hyperparameters of machine learning algorithms," *Journal of Machine Learning Research*, vol. 20, 2019.
- [11] M. Shakunthala and K. Helen Prabha, "Preprocessing analysis of brain images with atherosclerosis," *Proceedings of 2019 3rd IEEE International Conference on Electrical, Computer and Communication Technologies, ICECCT 2019*, 2019, doi: 10.1109/ICECCT.2019.8869009.
- [12] F. Novoselnik, R. Grbic, I. Galic, and F. Doric, "Automatic white blood cell detection and identification using convolutional neural network," *Proceedings of International Conference on Smart Systems and Technologies 2018, SST 2018*, pp. 163–167, 2018, doi: 10.1109/SST.2018.8564625.
- [13] K. Bousbai and M. Merah, "A comparative study of hand gestures recognition based on MobileNetV2 and ConvNet models," *Proceedings - 2019 6th International Conference on Image and Signal Processing and their Applications, ISPA 2019*, 2019, doi: 10.1109/ISPA48434.2019.8966918.
- [14] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018, doi: 10.1109/CVPR.2018.00474.
- [15] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, [Online]. Available: <http://arxiv.org/abs/1704.04861>.
- [16] R. Patel and A. Chaware, "Transfer learning with fine-tuned MobileNetV2 for diabetic retinopathy," *2020 International Conference for Emerging Technology, INCET 2020*, 2020, doi: 10.1109/INCET49848.2020.9154014.
- [17] C. Buiu, V. R. Dănilă, and C. N. Răduță, "MobileNetV2 ensemble for cervical precancerous lesions classification," *Processes*, vol. 8, no. 5, 2020, doi: 10.3390/PR8050595.
- [18] N. H. Barnouti, "Improve face recognition rate using different image pre-processing techniques," *American Journal of Engineering Research (AJER)*, vol. 5, no. 4, pp. 46–53, 2016, [Online]. Available: www.ajer.org.
- [19] S. Genc, K. N. Akpinar, and S. Karagol, "Automated abnormality classification of chest radiographs using MobileNetV2," *HORA 2020 - 2nd International Congress on Human-Computer Interaction, Optimization and Robotic Applications, Proceedings*, 2020, doi: 10.1109/HORA49412.2020.9152607.
- [20] K. Thenmozhi and U. S. Reddy, "Image processing techniques for insect shape detection in field crops," *Proceedings of the International Conference on Inventive Computing and Informatics, ICICI 2017*, pp. 699–704, 2018, doi: 10.1109/ICICI.2017.8365226.
- [21] J. Shijie, W. Ping, J. Peiyi, and H. Siping, "Research on data augmentation for image classification based on convolution neural networks," *Proceedings-2017 Chinese Automation Congress, CAC 2017*, vol. 2017-January, pp. 4165–4170, 2017, doi: 10.1109/CAC.2017.8243510.
- [22] C. C. Aggarwal, *Neural Networks and Deep Learning*. Cham: Springer International Publishing, 2018.
- [23] Z. Gao and X. Wang, "Deep learning," *EEG Signal Processing and Feature Extraction*, pp. 325–333, 2019, doi: 10.1007/978-981-13-9113-2_16.
- [24] G. Montavon, G. B. Orr, and K.-R. Mueller, "Neural networks: Tricks of the trade - Second Edition," *Springer*, pp. 639–655, 2012.
- [25] M. J. Macawile, V. V. Quiñones, A. Ballado, J. Dela Cruz, and M. V. Caya, "White blood cell classification and counting using convolutional neural network," *2018 3rd International Conference on Control and Robotics Engineering, ICCRE 2018*, pp. 259–263, 2018, doi: 10.1109/ICCRE.2018.8376476.

BIOGRAPHIES OF AUTHORS







Parmonangan R. Togatorop    Master's in information technology, Bachelor in information system, and Faculty Members & Researcher in Institut Teknologi Del. Research area in artificial intelligence, enterprise system, and data warehouse. She can be contacted at email: mona.togatorop@del.ac.id







Yohanssen Pratama    Current Faculty Members & Researcher in Institut Teknologi Del. 4+ years experience specializing in back-end/infrastructure, analytical tools development and computer programming. Teach academic and vocational subjects to undergraduates and also pursue my own research to contribute to the wider research activities of my department. He also a doctoral candidate in Nara Institute of Science and Technology and can be contacted at email: yohanssen.pratama@del.ac.id or yohanssen.pratama.yl0@is.naist.jp







Astri Monica Sianturi     Currently working as a Software Quality Assurance in PT. Phincon which is an IT consultant in Jakarta focusing on middleware and CRM. Have a professional experience of 2+ years in developing software and using development tools. She can be contacted at email: astrimsianturi@gmail.com



Mega Sari Pasaribu     Currently working as a Software Quality Assurance in PT. Phincon which is an IT consultant in Jakarta focusing on middleware and CRM. Have a professional experience of 2+ years in developing software and using development tools. She can be contacted at email: megaspasaribu@gmail.com



Pebri Sangmajadi Sinaga     Currently working as a Software Engineer in PT. Tokopedia specific in Engineering Productivity which is one of the biggest e-commerce in Indonesia. Have a professional experience of 2+ years in developing software and using development tools. He can be contacted at email: sangmajadis@gmail.com