

High performance of optimizers in deep learning for cloth patterns detection

Irma Amelia Dewi, Mahesa Atmawidya Negara Ekha Salawangi

Department of Informatics, Institut Teknologi Nasional Bandung, Bandung, Indonesia

Article Info

Article history:

Received Jun 8, 2022

Revised Sep 26, 2022

Accepted Oct 11, 2022

Keywords:

Clothing pattern

Convolutional neural network

Optimizer

ResNet

RetinaNet

ABSTRACT

In deep learning, optimization methods are an essential role. Optimizers are used to change weights and learn rates to reduce or minimize losses in a neural network. Nowadays, deep learning is widely used, especially in object detection tasks. In this case, cloth patterns are considered for object detection and assist visually impaired people. The visually impaired person had many limitations when doing their activity, not least when choosing clothes; it would be difficult without guidance or tools like Braille labels. In this research, a system was researched to detect 11 different cloth patterns (Argyle, Batik, Camouflage, Gingham, Dotted, Floral, Leopard, Solid, Striped, Zebra, and Zigzag) using RetinaNet with ResNet-152 architecture. To achieve the best model performance, compare 6 optimizers, such as Stochastic Gradient Descent, Root Mean Square Propagation, Adaptive Moment Estimation, Adaptive Delta, Adaptive Norm, and Adaptive Gradient was conducted. Each optimizer has trained with three different learning rates (1E-3, 1E-4, and 1E-5). A model with Adamax optimizer and learning rate 1E-4 was achieved with the highest accuracy with mAP (mean Average Precision) 91.28% during the training process. Based on the testing result this model was achieved precision 93.01%, recall 92.91%, F1-Score 92.79%, and accuracy 92.91%.

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



Corresponding Author:

Irma Amelia Dewi

Department of Informatics, Institut Teknologi Nasional Bandung

Bandung, Indonesia

Email: irma_amelia@itenas.ac.id

1. INTRODUCTION

Deep learning is a method of artificial neural network, which is also a branch of machine learning. Deep learning has been implemented in several domains, such as science, business, and government [1]. Deep learning has more than one hidden layer with its parameters. These parameters help predict a value from given input [2]. In deep learning and machine learning context, there are two types of parameters. First is a parameter that can be initialized and updated through the data learning process, named model parameters. While others are called hyperparameters, which require a tuning process to find the best and optimal configuration that gives the best results for a model. In deep learning, various hyperparameters can be tuned or changed manually, such as the number of hidden layers, units per layer, optimizer, activation function, learning rate, dropout rate, epochs, batch size, and early stop process [3].

One of the powerful algorithms in deep learning is convolutional neural network (CNN) which uses convolution processes. CNN is an effective algorithm to be used in object detection issues, and this learning algorithm also can be used in cloth pattern detection. Several variations of the CNN approach for object detection include region-based convolutional neural networks (RCNN), Fast-RCNN, Faster-RCNN, single-shot multibox detection (SSD), you only look once (YOLO), and RetinaNet. Although it is categorized as a

one-stage detector, it has been proven that RetinaNet speed has exceeded the one-stage detector, with accuracy above the average of the two-stage detector. RetinaNet also has overcome the imbalanced data issue with the focal loss technique and feature pyramid network (FPN) to detect small objects in an image. One of the best CNN architecture is residual network (ResNet) which won ImageNet large scale visual recognition challenge (ILSVRC) in 2015. Several studies discussing the combination of RetinaNet and residual network have shown significant results [4]–[6] both in training and testing. ResNet implemented the concept of residual learning or also known as skip connections. ResNet has several types of architecture, such as ResNet-34, ResNet-101, and ResNet-152. Although ResNet has many layers, the complexity is lower than other architectures such as visual geometry group network (VGGNet) [4]. However, the architecture itself cannot guarantee to get the best performance in all cases. In CNN specifically, algorithm tuning or hyperparameter tuning is done manually to find the best performance for a specific case. One of the hyperparameters is called an optimizer.

Several studies comparing different state-of-the-art gradient descent-based optimizers have been conducted. Prilianti *et al.* [5] proposed a system to predict photosynthetic pigments from multispectral plant digital image input by utilizing a CNN. To observe the best fit for their case, algorithm tuning such as architecture, epoch, and optimizer was conducted. For comparison, the paper used three different architectures (ShallowNet, LeNet, and AlexNet), four epochs (60, 75, 90, and 105), and seven optimizers (Stochastic gradient descent (SGD), adaptive gradients (Adagrad), adaptive learning rate method (Adadelata), root mean square propagation (RMSProp), adaptive moment estimation (Adam), Adamax, and the nesterov-accelerated adaptive moment estimation (Nadam)). From their experimental results, it is shown that LeNet performs consistently small in mean squared error (MSE), and across the 7 optimizers, adam can maintain low MSE in the next epoch after it reaches the minimum on a particular epoch. Dogo *et al.* [6] discuss a comparative analysis of gradient descent algorithms in a simple CNN architecture. There are three different popular classification problems: cat and dog classification, Fashion, modified national institute of standards and technology (MNIST), and natural images (Airplane, cat, dog, people, flower and fruit). The research was done by comparing seven optimizers (SGD, RMSProp, Adam, Adagrad, Adadelata, Adamax, and Nadam) in each mentioned dataset. The paper concludes that each optimizer performed varied results in each dataset, and Nadam shows a superior and robust optimizer across three different datasets than the other optimization techniques. Yaqub *et al.* [7] built a brain tumor classification and segmentation system using their proposed CNN architecture. The paper used magnetic resonance imaging (MRI) images obtained from the public brain image dataset to train on 10 different state-of-the-art gradient descent-based optimizers, such as Adagrad, Adadelata, SGD, Adam, cyclical learning rates (CLR), Adamax RMSProp, Nadam, and Nesterov accelerated gradient (NAG). From the training results across 10 different optimizers, Adam outperformed other optimizers with the highest accuracy of 99.2%. Although all optimizers performed consistent results in both quantitative and graphical, adam still performs much better with the lowest error rate when it reaches the minimum in a particular epoch.

On the other side, cloth pattern classification research has been conducted by Stearns *et al.* [8]. The paper discussed applying transfer learning to convolutional neural networks with ResNet-101 architecture pre-trained on the ImageNet object dataset. The dataset consists of 6 classes of cloth pattern classes such as solid, striped, checkered, dotted, zigzag, and floral. Although the dataset is relatively small, they implement data augmentation to enrich their dataset and randomly set 6,400 images to the training set and 1,600 images to the test set. The classification accuracy on the test set achieved 91.7% from the training results. However, when it included variations of distance, rotation, perspective, and fabric tension, the accuracy dropped to 72.8%. Rasyidi and Bariyah [9] analyze research about batik recognition using a CNN. In this study, CNN was applied to identify 6 different batik patterns, such as Banji, Ceplok, Kawung, Mega Mendung, Parang, and Sekar Jagad, with an amount of 994 images In total and divided into 80% training set and 20% test set. To achieve the best results, this paper compared the performance of the following CNN architectures and their variations: AlexNet, VGG (VGG16 and VGG19), ResNet (ResNet18, ResNet34, ResNet50, ResNet101, and ResNet152), SqueezeNet (SqueezeNet10 and szqueezeNet11), and DenseNet (DenseNet121, DenseNet169, DenseNet201, and DensNet161). The experimental results on the test data showed that CNN produced an excellent performance by using DenseNet network architecture with an accuracy of 94% and top-2 accuracy of 99%. Hussain *et al.* [10] built an automated system for woven fabric classification and recognition to improve productivity. The paper proposed a deep learning model by applying the data augmentation and transfer learning approaches; the residual model network was used to support the training process. Besides that, a comparative analysis was carried out with other baseline approaches and compared with the VGGNet pre-trained model. The several parameters that have been evaluated show that the proposed method performed better than the other existing studies with a precision of 98.3%, recall of 99.1%, and accuracy of 99.3%.

As mentioned in [5]–[7], it is shown that all optimizers were not performing consistently well in all cases. Although Adam has shown a significant result in most cases, it cannot be ensured to be always the best optimizer, particularly in the cloth pattern detection issue. Therefore, research for comparing optimizers in cloth pattern detection is still needed to find the best fit result. Unfortunately, in [8], there were in total only six classes about cloth patterns, and the mentioned paper [5]–[7] only conducted the comparison between CNN architectures, which only perform object classification. Therefore, the number of clothing patterns detected was 11 classes: argyle, batik, camouflage, gingham, dotted, floral, leopard, solid, striped, zebra, and zigzag. The CNN architecture used in this research is RetinaNet with ResNet 152 backbone. To get the highest performance in this study, a comparison of 6 optimizer methods was conducted: SGD, RMSProp, Adam, Adadelta, Adamax, and Adagrad in 3 different learning rates (1E-3, 1E-4, 1E-5).

2. METHOD

This section described the proposed cloth pattern detection methodology using RetinaNet and ResNet-152 architecture. Deep learning requires a large dataset of images. In this case, there are 11 types of cloth patterns such as argyle, batik, camouflage, gingham, dotted, floral, leopard, solid, striped, zebra, and zigzag. The processes of the system are illustrated in Figure 1.

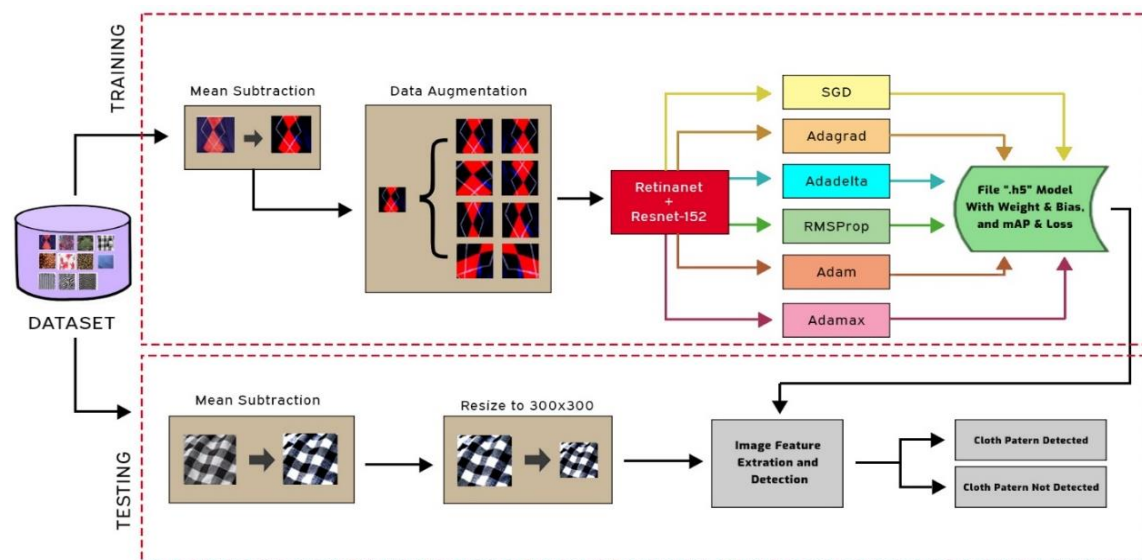


Figure 1. Block diagram cloth pattern detection

Before the dataset is sent to the model to train, 2 data preparation processes are conducted, both of them are mean subtraction and data augmentation. Mean subtraction is done to normalize the dataset, then to enrich the dataset, data augmentation is performed. After the dataset is prepared, it was sent to the program to execute the training process. The training process is conducted in each optimizer and each learning rate. The training process generated a model with a detailed graph about accuracy and losses. The gained accuracy and losses were compared to each model to find the best fit model for cloth patterns detection.

2.1. Dataset

In deep learning, data is needed in massive amounts, as it was be divided into good and bad samples [1], [11], [12]. In this case, the cloth patterns dataset has been collected from many sources, such as the The City College of New York (CCNY) dataset, data.world.com website, Google image dataset, datasets from public repositories, and images captured from SQ11 mini camera. Figures 2(a)-2(k) shows an example of 11 different cloth patterns used in this research.

The amount of dataset collected is 12,498 in total and divided into 10,983 training set (1,019 argyle, 953 batik, 959 camouflage, 1,015 gingham, 1,011 dotted, 1,023 floral, 928 leopards, 1,026 solid, 1,023 striped, 1,014 zebra, and 1,012 zigzag), 965 validation set (90 argyles, 86 batik, 90 camouflage, 89 gingham, 90 dotted, 90 floral, 70 leopards, 90 solid, 90 striped, 90 zebra, and 90 zigzag), and 550 test set (50 in each cloth pattern class).

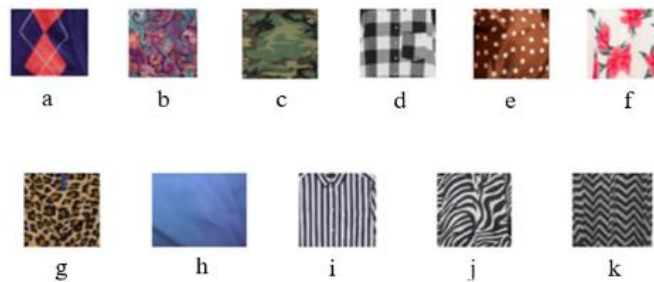


Figure 2. Example image in each cloth patterns class; (a) Argyle, (b) Batik, (c) Camouflage, (d) Gingham, (e) Dotted, (f) Floral, (g) Leopard, (h) Solid, (i) Striped, (j) Zebra, and (k) Zigzag

2.2. Mean subtraction (Normalization)

Gaining more accuracy and reducing losses primarily used a method called normalization [13]. In the image context, the mean of the image subtracts the value of a pixel from each channel. Normalization itself is a process to remove the difference in magnitude between different features, which benefits learning [14]. Normalization can also help the model prevent overfitting [15], [16]. Aside from accuracy and loss, it can also accelerate the training process as it keeps important feature maps and removes or reduces unwanted parts of pixels. Figure 3(a) is the original image implementing mean subtraction in the argyle cloth pattern as shown in Figure 3(b).

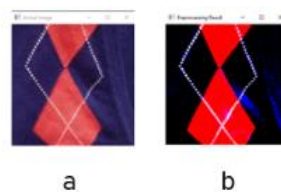


Figure 3. Mean subtraction example of cloth pattern; (a) Original image and (b) The result from the mean subtraction process

2.3. Data augmentation

In deep learning, an immense amount of data is required to achieve the best result. However, collecting and gathering a large dataset is time-consuming, even though it's hard to find unique data differentiating from others. Data augmentation generates unique and diverse data to enhance the size and quality of training datasets [17]. The data augmentation process consists of rotation, translation, scaling, and flipping in this case as shown in Figure 4(a) which is the original image and Figures 4(b)-4(h) are the results of the augmented image.

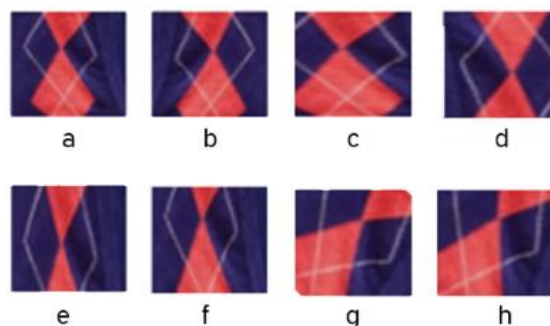


Figure 4. Data augmentation example on cloth pattern; (a) Original image, (b) Flip horizontal, (c) Scale Horizontal, (d) Flip vertical, (e) Scale horizontal to smaller size, (f) Scale horizontal to a smaller size, (g) Flip vertical, and (h) Rotate

2.4. Optimizers

The optimization algorithm is needed to reduce the losses and gain accuracy. This algorithm can help the network find the local minimum error and update the internal parameters iteratively. One of the common choices for optimization algorithms is Standard Gradient Descent, which minimizes the error function during the training process [5]. To reduce error, during the training process, the parameters were updated in a reverse direction, then at each iteration desired output and predicted output will be compared, and the error is back-propagated [18]. Unfortunately, standard gradient descent is computationally expensive and requires large memory. Several optimizers improve common gradient descent, such as SGD, RMSprop, Adam, Adadelta, Adamax, and Adagrad.

2.4.1. Stochastic gradient descent (SGD)

SGD is a modification from standard gradient descent that reduces the computational level. Instead of updating the parameter in the entire dataset, SGD uses a subset of training data to update the parameters; therefore, the SGD optimizer requires less memory [19]. SGD weight update rule is described in (1).

$$\theta_{t+1} = \theta_t - \eta d_t \quad (1)$$

where:

θ = Update Parameter

t = time step

η = learning rate

d_t = gradient of objective function based on θ

2.4.2. Adaptive gradient (Adagrad)

Stochastic gradient descent used the same learning rate for updating the parameters. Adagrad offered to change the learning rate at each time step regarding the importance of parameters update [5]. However, Adagrad also has several disadvantages, it's computationally expensive as it needs to calculate second-order derivative, and as the learning rate decreases, it results in slow training, but the algorithm provides an approximate minimization, the speed of its convergence, this makes it possible to use it in future work using large data sets [20]. In (2) shows the weight update rule for Adagrad.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{D_t + \epsilon}} d_t \quad (2)$$

Where:

D_t = Diagonal matrix

ϵ = Vector of small numbers to avoid division by zero

2.4.3. Adaptive delta (Adadelta)

Adadelta is a robust extension of Adagrad, which has overcome the decaying learning rate issue. Instead of accumulating previously squared gradients, the idea of Adadelta is limiting the accumulating process so that it accumulates only for w squared gradients [21]. Since in $(RMS)[d]_t$, the parameter was be approximated to update until the previous time step. Hence the learning rate η replace with $(RMS)[\Delta\theta]_{t-1}$ as the previous time. In (6) described the final formula of the weight update rule for Adadelta.

$$\Delta\theta_t = - \frac{\eta}{(RMS)[d]_t} d_t \quad (3)$$

$$\Delta\theta_t = - \frac{(RMS)[\Delta\theta]_{t-1}}{(RMS)[d]_t} d_t \quad (4)$$

$$\theta_{t+1} = \theta_t - \Delta\theta_t \quad (5)$$

$$(RMS)[d]_t = \sqrt{E[\Delta\theta^2]_t + \epsilon} \quad (6)$$

Where:

$E[\Delta\theta^2]_t$ = Decaying average over past squared gradients

2.4.4. Root mean squared propagation (RMSProp)

RMSProp is a published method for Adaptive learning rate found by Geoff Hinton. Like Adadelta, RMSProp was created to solve the Adagrad issue, which overcomes the decaying learning rate problem [19]. Instead of accumulating all of the gradients for momentum, it defined a fixed window for accumulating gradients as described in (7).

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{R[d^2]_t + \epsilon}} d_t \quad (7)$$

Where,

$R[d^2]_t$ = Exponentially decaying average of squared gradients

2.4.5. Adaptive momentum (Adam)

Adam is a method for efficient stochastic optimization, as it requires less memory for computation processes. The idea of Adam is to combine two popular methods, Adagrad, which works well with sparse gradients, and RMSProp, which works well in online and non-stationary settings [22], [23]. In (8) shows the formula of the weight update rule for the Adam optimizer.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{u}_t + \epsilon}} \hat{a}_t \quad (8)$$

Where,

$$\hat{a}_t = \frac{a_t}{1 - \beta_1^t} \quad (9)$$

$$\hat{u}_t = \frac{u_t}{1 - \beta_2^t} \quad (10)$$

Detailed (9) and (10) information:

a_t = exponentially decaying average of past gradients

u_t = exponentially decaying average of past squared gradients

β_2 = Value near zero

2.4.6. Adaptive max pooling (Adamax)

Adamax is an extension of Adam's optimizer, which is sometimes superior compared to Adam. The idea of this Adam's modification is the use of the infinity norm [5] This indicates that if AdaMax is preferred as the optimization algorithm, there is no need to correct for initialization bias. In addition, the magnitude of the parameter update has a more superficial bond than Adam [24]. In (11) illustrated the weight update rule for the Adamax optimizer.

$$\theta_{t+1} = \theta_t - \frac{\eta}{v_t} \hat{a}_t \quad (11)$$

Where:

v_t = Infinity norm

2.5. RetinaNet and ResNet-152

The object detection algorithm RetinaNet used feature pyramid network (FPN) concept with focal loss to compare optimizers. RetinaNet has proven that it outperformed the previous object detection algorithm [25], in both speed (Exceeded one-stage detector) and accuracy (Exceeded two-stage detector). At the same time, focal loss is used to taking care of imbalanced data issues [26]–[28]. As illustrated in Figure 5(a), the construction of FPN involves a bottom-up pathway, which takes care of feed-forward computation of the backbone ConvNet and computes feature hierarchy consisting of feature maps, in this process ResNet [4] architecture was used. The top-down pathway upsampling spatially coarser features maps, enhanced with features from the bottom-up pathway via lateral connections as shown in Figure 5(b) [25], [29].

A residual neural network or known as ResNet, was chosen as a backbone for RetinaNet, the architecture won the ILSVRC competition in 2015 with an error rate achieved of 3.57% [4] which has exceeded the human level. By applying the concept of residual learning or skip connection, ResNet manages to lower the computation. Although ResNet has many layers, the ResNet computation level is lower than

VGG, whose layer is less than ResNet. Across all ResNet types, ResNet-152 was chosen because it outperformed the other architecture with top-5 err achieved 4.49% [4]. Figure 6 illustrates the concept of residual network or skip connections.

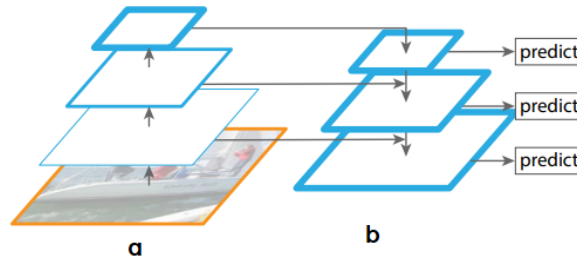


Figure 5. Illustration of feature pyramid network (FPN) in; (a) Bottom-up pathway: the backbone network (ResNet-152), and (b) Top-down pathway: upsamples spatial coarser feature maps from higher pyramid level and lateral connection: merge the bottom-up layer with a top-down layer [25]

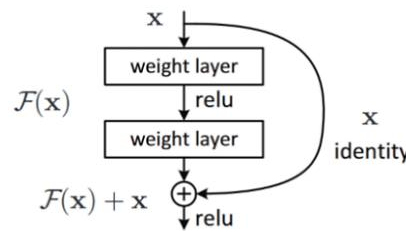


Figure 6. Illustration of skip connection concept in residual network [4]

2.5. Testing process

From each mAP obtained by a model after training processes, several parameters are considered for comparing the performance evaluation metrics of each model, such as precision, recall, F1-Score, and accuracy.

Detailed (12)-(14) information:

- $\sum TP_{Class}$ = Data with correct prediction in class positive (True positive)
- $\sum FP_{Class}$ = Data with the wrong predictions in the class positive (False positive)
- $\sum FN_{Class}$ = Data with the wrong predictions in the class negative (False negative)
- n = Total Data

$$\text{Accuracy} = \frac{\sum TP_{Class}}{n} \quad (12)$$

$$\text{Precision} = \frac{\sum TP_{Class}}{\sum TP_{Class} + \sum FP_{Class}} \quad (13)$$

$$\text{Recall} = \frac{\sum TP_{Class}}{\sum TP_{Class} + \sum FN_{Class}} \quad (14)$$

3. RESULTS AND DISCUSSION

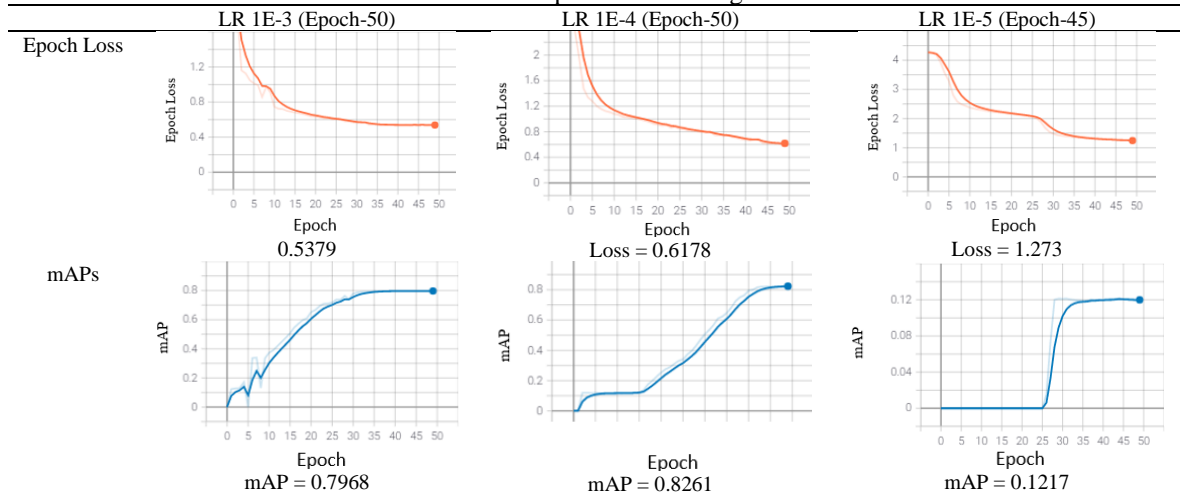
This section explains the result obtained from the training and testing process for six different optimizers and three learning rates, the mAP, and loss. The result from the training process would be described and illustrated as a graph. The average accuracy score obtained in the testing process is briefly discussed.

3.1. Training result

The training process has been conducted in each optimizer, and each learning rate, accuracy, and loss for each model were obtained and described in Tables 1-6. The training process used several hyperparameters; batch size (1), steps (4,858), epoch (50), and six different optimizers along with 3 different

learning rates. Approximately 775 seconds were taken during the training process for an epoch, with the maximum size of the image is 1,333 in width and 800 in height. The experiment results in the SGD optimizer shows that learning rate 1E-3 and 1E-4 obtained high mAP. Unfortunately, the opposite for learning rate 1E-5 which obtained low mAP with high loss. From the graphic illustrated in Table 1, learning rate 1E-5 still can increase mAP and decrease the loss with a higher epoch.

Table 1. SGD optimizers training result



While in RMSprop optimizer models, it's inversely proportional to the previous model, where learning rates 1E-3 and 1E-4 show significantly low mAP scores. Learning rate 1E-3 still has a possible increase in the mAP, unfortunately learning rate 1E-4 indicates overfitting. On the other hand, learning rate 1E-5 has successfully obtained a high mAP score with an accuracy of 88.82%, as shown in Table 2.

Adam optimizer performed great in learning rate 1E-4 with 81.33% and 1E-5 with 85.48% accuracy. While at learning rate 1E-3, the model starts overfitting from epoch 15, as shown in Table 3. As shown in Table 4, from the training process with the Adadelta optimizer, the only model that shows an excellent result is learning rate 1E-3, while the rest might have a possibility to gain more mAP and reduce model loss with a higher epoch.

A model with an Adamax optimizer produced the highest average of mAP compared with the other five optimizers. Although the learning rate 1E-3 mAP is the lowest, getting a higher mAP with a higher epoch is still possible. The best model attained in this experiment is a model with Adamax optimizer and learning rate 1E-4 with an accuracy of 91.28%, which have a slightly different value than learning rate 1E-5, as shown in Table 5.

Table 2. RMSProp optimizer training result

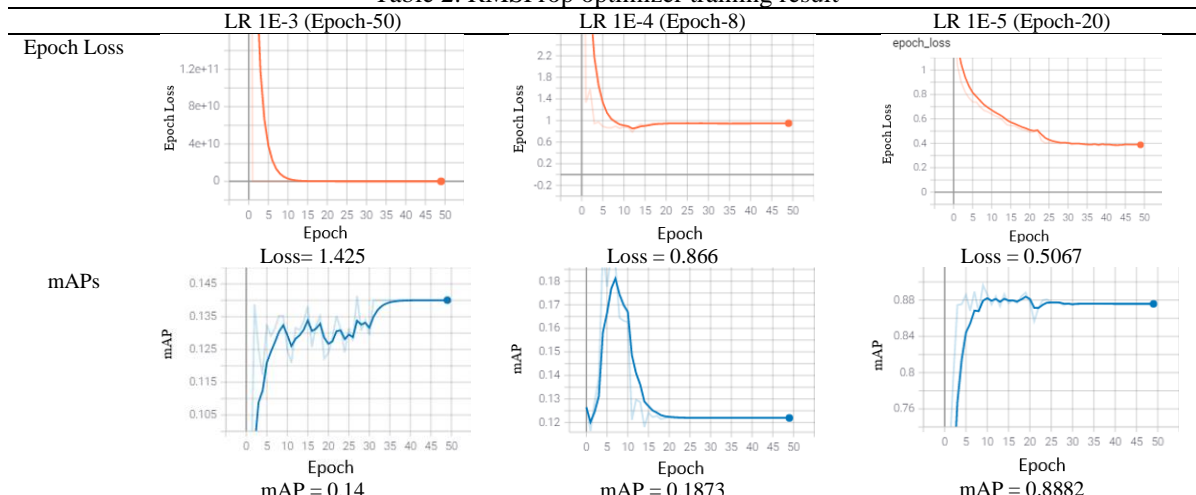


Table 3. Adam optimizer training result

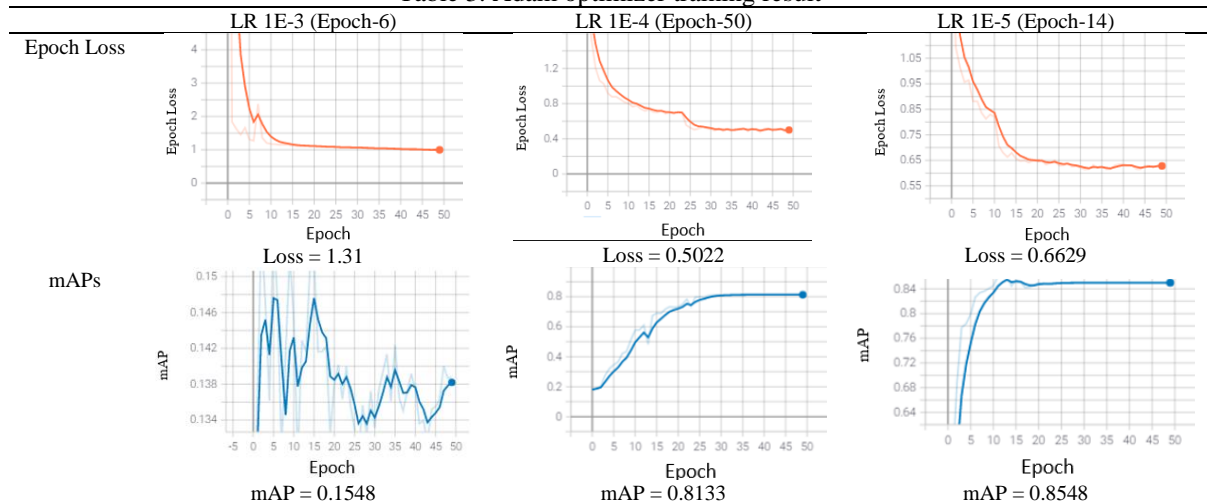


Table 4. Adadelata optimizer training result

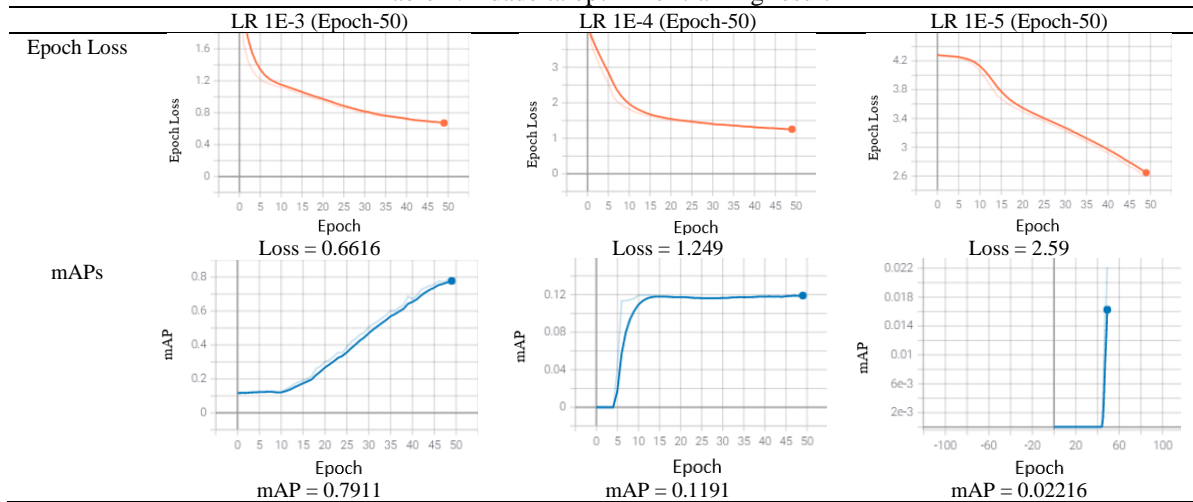
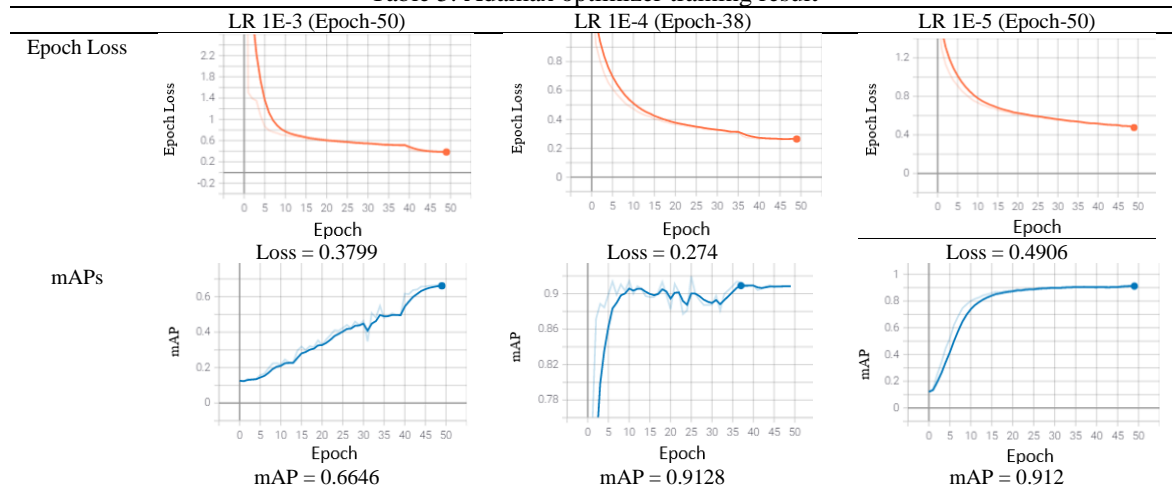


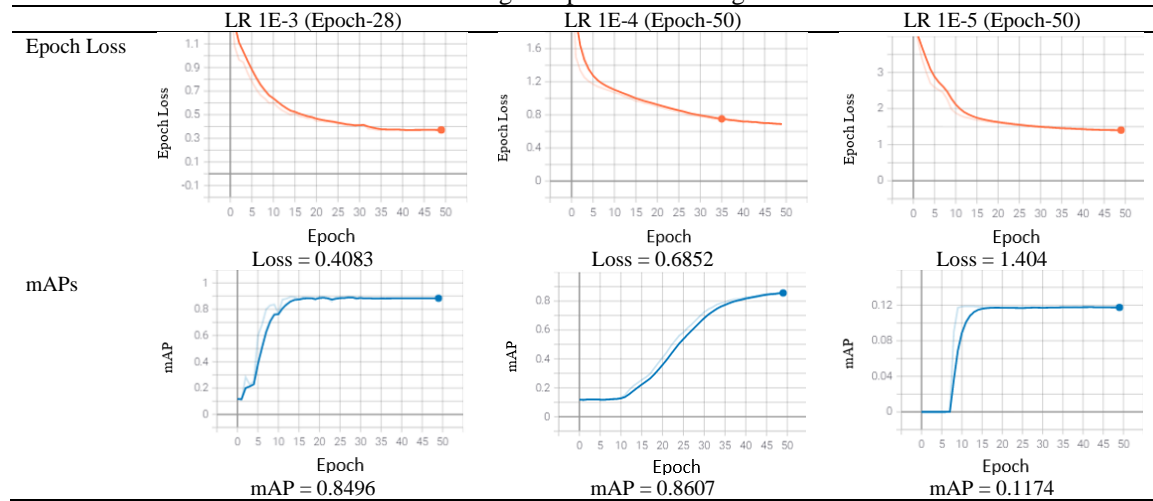
Table 5. Adamax optimizer training result



Adagrad optimizer performs similarly to SGD optimizer, where both learning rates 1E-3 and 1E-4 show significantly higher scores with 84.96% and 86.07%. The model with the Adagrad optimizer, learning

rate 1E-5 gets a relatively low mAP value for each clothing pattern. The graph illustrated in Table 6 shows that learning rate 1E-5 can reduce the loss with a higher epoch.

Table 6. Adagrad optimizer training result



The data gathered in Table 7 shows that there is no single optimizer that can consistently give the best result for all cases [5], and the learning rate hyperparameter played a significant role in model performance. In this case, the best optimizer for detecting cloth patterns is a model with an Adamax optimizer with a learning rate of 1E-4 with an accuracy of 91.28%. The training result obtained shows that each optimizer does not consistently achieve a good result, and the learning rate affects the training result [30], [31].

Table 7. mAP result from each optimizer and learning rate

Learning Rate	Optimizer					
	SGD	RMSprop	Adam	Adadelata	Adamax	Adagrad
1E-3	79.68%	14%	15.64%	79.11%	66.46%	84.96%
1E-4	82.61%	18.73%	81.33%	0%	91.28%	86.07%
1E-5	12.17%	88.82%	85.48%	2.216%	91.2%	11.74%

3.2. Testing results

For comparison between optimizer and learning rate, parameters like precision, recall and F1-Score were taken into consideration. Table 8 provided the calculation result of precision, recall, and F1-Score from 10 models that have been tested. For testing purposes, only 10 models be tested, the model chosen for the testing process is a model with mAP above 70%, while the rest of the models were ignored. The highest score was obtained in each cloth pattern (Argyle, Batik, Camouflage, Gingham, Dotted, Floral, Leopard, Solid, Striped, Zebra, Zigzag) for better visualization as green. As shown in Table 8, Adamax with a learning rate of 1E-4, outperformed other models with five cloth patterns: batik, camouflage, gingham, zebra, and zigzag. The average precision for all class cloth patterns using Adamax optimizer and 1e-4 learning rate is 93.01 %. Optimizer Adamax and learning rate 1E-4 still surpassed other models with an average recall of 92.91%. F1-Score is a calculation for the conclusion of precision and recall. From Table 8, it is seen that models with optimizer Adamax and learning rate 1E-4 have 3 achieved the highest score with an average in F1-Score of 92.79%.

Table 8. Precision in each pattern and each optimizer

	Adadelata 1E-3	Adagrad 1E-3	Adagrad 1E-4	Adam 1E-4	Adam 1E-5	Adamax 1E-4	Adamax 1E-5	RMSProp 1E-5	SGD 1E-3	SGD 1E-4
Precision	75.82%	91.19%	78.96%	86.34%	84.42%	93.01%	91.72%	91.43%	68.04%	68.79%
Recall	76.00%	90.73%	79.09%	86.00%	84.55%	92.91%	91.64%	91.45%	70.91%	68.91%
F1-Score	75.80%	90.73%	78.88%	85.90%	84.41%	92.79%	91.62%	91.40%	69.52%	68.76%

Figure 7 shows that a model with Adamax optimizer with learning rate 1E-4 obtained the best result among other models with an accuracy of 92.91%. The Adamax model has a higher probability of getting mAP when the Epoch value is added. Unfortunately, 2 models from the SGD optimizer performed low results with accuracy of 68.91% (Learning rate 1E-4) and 69.27% (Learning rate 1E-3).

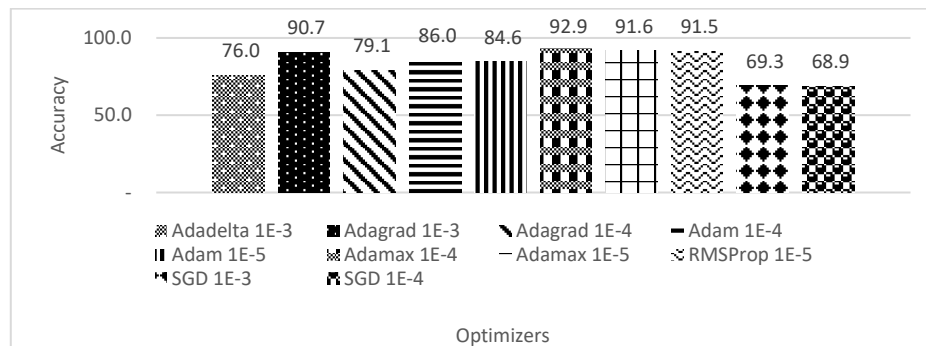


Figure 7. Comparison graph of system performance accuracy based on the optimizers and learning rates

4. CONCLUSION

The experimental result compared six different optimizers with three different learning rates. Both in training and testing results, the model with Adamax optimizer and learning rate 1E-4 proved to be the best result for the cloth pattern detection task with mAP of 91.28% in the training result. In the testing result, from 4 parameters taken into consideration, it is shown that the model outperformed in precision at 93.01 %, recall at 92.91%, F1-Score at 92.79%, and accuracy at 92.91%. While the model with SGD optimizer seems to be the worst model for this particular task, with mAP of 79.68% (Learning rate 1E-3) in the training result and accuracy of 68.91% (Learning rate 1E-4) in the testing result.

ACKNOWLEDGEMENTS

The publishing of this article was supported by funding from the Institut Teknologi Nasional Bandung. The research equipment was supported and prepared by the laboratory of the department of informatics, Institut Teknologi Nasional Bandung.




REFERENCES

- [1] K. Horak and R. Sablatnig, "Deep learning concepts and datasets for image recognition: Overview 2019," p. 100, 2019, doi: 10.1117/12.2539806.
- [2] J. Brownlee, "How to configure the number of layers and nodes in a neural network," *Better Deep Learning*, pp. 1–20, 2018, [Online]. Available: <https://machinelearningmastery.com/how-to-configure-the-number-of-layers-and-nodes-in-a-neural-network/>.
- [3] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, 2020, doi: 10.1016/j.neucom.2020.07.061.
- [4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, pp. 770–778, 2016, doi: 10.1109/CVPR.2016.90.
- [5] K. R. Prilianti, T. H. P. Brotsudarmo, S. Anam, and A. Suryanto, "Performance comparison of the convolutional neural network optimizer for photosynthetic pigments prediction on plant digital image," *AIP Conference Proceedings*, vol. 2084, 2019, doi: 10.1063/1.5094284.
- [6] E. M. Dogo, O. J. Afolabi, N. I. Nwulu, B. Twala, and C. O. Aigbavboa, "A comparative analysis of gradient descent-based optimization algorithms on convolutional neural networks," *Proceedings of the International Conference on Computational Techniques, Electronics and Mechanical Systems, CTEMS 2018*, pp. 92–99, 2018, doi: 10.1109/CTEMS.2018.8769211.
- [7] M. Yaqub *et al.*, "State-of-the-art CNN optimizer for brain tumor segmentation in magnetic resonance images," *Brain Sciences*, vol. 10, no. 7, pp. 1–19, 2020, doi: 10.3390/brainsci10070427.
- [8] L. Stearns, L. Findlater, and J. E. Froehlich, "Applying transfer learning to recognize clothing patterns using a finger-mounted camera," *ASSETS 2018 - Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility*, pp. 349–351, 2018, doi: 10.1145/3234695.3241015.
- [9] M. A. Rasyidi and T. Bariyah, "Batik pattern recognition using convolutional neural network," *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 4, pp. 1430–1437, 2020, doi: 10.11591/eei.v9i4.2385.
- [10] M. A. I. Hussain, B. Khan, Z. Wang, and S. Ding, "Woven fabric pattern recognition and classification based on deep convolutional neural networks," *Electronics (Switzerland)*, vol. 9, no. 6, pp. 1–12, 2020, doi: 10.3390/electronics9061048.
- [11] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015, doi: 10.1038/nature14539.
- [12] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, "Deep learning applications and challenges in big data analytics," *Journal of Big Data*, vol. 2, no. 1, 2015, doi: 10.1186/s40537-014-0007-7.




- [13] J. Sun, X. Cao, H. Liang, W. Huang, Z. Chen, and Z. Li, "New interpretations of normalization methods in deep learning," *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence*, pp. 5875–5882, 2020, doi: 10.1609/aaai.v34i04.6046.
- [14] L. Huang, J. Qin, Y. Zhou, F. Zhu, L. Liu, and L. Shao, "Normalization techniques in training DNNs: Methodology, analysis and application," 2020, [Online]. Available: <http://arxiv.org/abs/2009.12836>.
- [15] C. F. G. Dos Santos and J. P. Papa, "Avoiding overfitting: A survey on regularization methods for convolutional neural networks," *ACM Computing Surveys*, vol. 54, no. 10s, pp. 1–25, 2022, doi: 10.1145/3510413.
- [16] C. Garbin, X. Zhu, and O. Marques, "Dropout vs. batch normalization: an empirical study of their impact to deep learning," *Multimedia Tools and Applications*, vol. 79, no. 19–20, pp. 12777–12815, 2020, doi: 10.1007/s11042-019-08453-9.
- [17] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, 2019, doi: 10.1186/s40537-019-0197-0.
- [18] S. Bera and V. K. Shrivastava, "Analysis of various optimizers on deep convolutional neural network model in the application of hyperspectral remote sensing image classification," *International Journal of Remote Sensing*, vol. 41, no. 7, pp. 2664–2683, 2020, doi: 10.1080/01431161.2019.1694725.
- [19] S. Ruder, "An overview of gradient descent optimization algorithms," 2016, [Online]. Available: <http://arxiv.org/abs/1609.04747>.
- [20] A. Mustapha, L. Mohamed, and K. Ali, "Comparative study of optimization techniques in deep learning: Application in the ophthalmology field," *Journal of Physics: Conference Series*, vol. 1743, no. 1, 2021, doi: 10.1088/1742-6596/1743/1/012002.
- [21] M. D. Zeiler, "ADADELTA: An adaptive learning rate method," 2012, [Online]. Available: <http://arxiv.org/abs/1212.5701>.
- [22] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.
- [23] R. Zaheer and H. Shaziya, "A study of the optimization algorithms in deep learning," *Proceedings of the 3rd International Conference on Inventive Systems and Control, ICISC 2019*, pp. 536–539, 2019, doi: 10.1109/ICISC44355.2019.9036442.
- [24] D. Soydaner, "A comparison of optimization algorithms for deep learning," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 34, no. 13, 2020, doi: 10.1142/S0218001420520138.
- [25] X. Li, T. Lai, S. Wang, Q. Chen, C. Yang, and R. Chen, "Weighted feature pyramid networks for object detection," *Proceedings - 2019 IEEE Intl Conf on Parallel and Distributed Processing with Applications, Big Data and Cloud Computing, Sustainable Computing and Communications, Social Computing and Networking, ISPA/BDCloud/SustainCom/SocialCom 2019*, pp. 1500–1504, 2019, doi: 10.1109/ISPA-BDCloud-SustainCom-SocialCom48970.2019.00217.
- [26] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 318–327, 2020, doi: 10.1109/TPAMI.2018.2858826.
- [27] N. N. Nguyen and A. T. Duong, "Comparison of two main approaches for handling imbalanced data in churn prediction problem," *Journal of Advances in Information Technology*, vol. 12, no. 1, pp. 29–35, 2021, doi: 10.12720/jait.12.1.29-35.
- [28] M. M. Sheikh, F. R. Sayem, and M. A. R. Ahad, "A residual network with focal loss to handle class-imbalance problem on nurse care activity recognition," *2021 Joint 10th International Conference on Informatics, Electronics and Vision, ICIEV 2021 and 2021 5th International Conference on Imaging, Vision and Pattern Recognition, icIVPR 2021*, 2021, doi: 10.1109/ICIEVICIVPR52578.2021.9564165.
- [29] T. Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, pp. 936–944, 2017, doi: 10.1109/CVPR.2017.106.
- [30] A. J. e D. P. P. S. Rathore, N. Dadich, "Effect of learning rate on neural network and convolutional neural network," *International Journal of Engineering Research and Technology (IJERT)*, vol. 06, no. 17, pp. 1–8, 2018, doi: 10.17577/IJERTCONV6IS17007.
- [31] J. Jepkoech, D. M. Mugo, B. K. Kenduiywo, and E. C. Too, "The effect of adaptive learning rate on the accuracy of neural networks," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 8, pp. 736–751, 2021, doi: 10.14569/IJACSA.2021.0120885.

BIOGRAPHIES OF AUTHORS



Irma Amelia Dewi    received a B.Sc. on Informatics Engineering from Institut Teknologi Nasional Bandung, Indonesia in 2010 and a M.Sc. on Computer Engineering from Institut Teknologi Bandung (ITB), Indonesia, in 2013. Her research interests are Digital Image Processing, Computer Vision, and Artificial Intelligence. She works as a lecturer in the Department of Informatics, Institut Teknologi Nasional, Itenas, Bandung. She can be contacted at email: irma_amelia@itenas.ac.id.



Mahesa Atmawidya Negara Ekha Salawangi    received a B.Sc. on Informatics Engineering from Institut Teknologi Nasional Bandung, Indonesia in 2021. His research interests are Artificial Intelligence and software development. He works as an employee in Weekend Inc, a digital product agency in Jakarta, Indonesia. He can be contacted at email: mahesaatmanegaraekasalawangi@gmail.com.