Journal of Engineering Science and Technology
Vol. 18, No. 2 (2023) 1113 - 1123

IMPLEMENTATION OF CONVOLUTIONAL NEURAL NETWORK FOR SUNDANESE SCRIPT HANDWRITING RECOGNITION WITH DATA AUGMENTATION

IRFAN MALIKI*, ADE SYAHLAN PRAYOGA

Department of Informatics Engineering, Faculty of Engineering and Computer Science, Universitas Komputer Indonesia, Jl. Dipatiukur no 112-116 Bandung 40132 Indonesia
*Corresponding Author: irfan.maliki@email.unikom.ac.id

Abstract

Sundanese script is one of the cultural heritages that need to be preserved. However, Sundanese script has complexity and uniqueness in its writing, making it difficult to recognize. The recognition can be done automatically using deep learning. One of the problems is that the recognition has a small amount of data and is less varied. In this study, the proposed solution is to use data augmentation. This study focused on how the use of data augmentation can help to improve accuracy in performing the recognition of handwriting image pattern using Convolutional Neural Network (CNN) method. Data augmentation is the process of artificially increasing the amount of data by generating new data points from existing data. The augmentation includes adding small changes to the data or using machine learning models to generate new data points in the latent space of the original data in order to strengthen the data set. Data augmentation applied in this research is flipping, rotation, and translation techniques. Based on the results, it can be concluded that the use of data augmentation to increase the number and variety of data samples has a significant effect on the accuracy value of 0.1707 or about 17.07%. The best accuracy obtained is 0.8 or 80% using a baseline model with data augmentation. These findings yielded good results because the system is able to perform image classification quite well.

Keywords: Convolutional neural network, Data augmentation, Handwriting recognition, Sundanese script.

1. Introduction

Sundanese script was used by the Sundanese tribe in the past to communicate [1]. The complexity and uniqueness of the Sundanese character makes this script difficult in its recognition [2]. Character recognition from Sundanese handwriting requires special knowledge to be able to understand the meaning of each character. Therefore, the introduction of Sundanese script can be developed through activities related to computerization, one of which is by conducting research related to pattern recognition of Sundanese handwriting imagery [3, 4].

Previous research proposed a method for Sundanese character recognition by combining Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks. The proposed method achieved an accuracy of 94.22% on a dataset of 5,000 Sundanese characters [5]. Another research work proposed a hybrid approach to recognize Sundanese characters, which combines deep learning approach and Support Vector Machines (SVMs). The proposed method achieved an accuracy of 91.03% on a dataset of 1,000 Sundanese characters [6]. Based on several previous research, the application of some methods for image pattern recognition of Sundanese handwriting received imperfect results or there are still several unclassified characters [7-15]. However, in previous research, CNN method receives the best accuracy compared to several other previous methods such as network artificial nerves, Modified Direction Feature (MDF), and Learning Vector Quantization (LVQ). CNN's methods yielded good results regarding the introduction of handwriting patterns. However, in previous research, the CNN method also has disadvantages when trained because of the lack of data sample and variance so that the accuracy obtained less maxim.

The method: data samples and unvaried data can affect the performance of the CNN method; therefore, the use of general data augmentation was done to increase image variation by manipulating the image transformation [16-18]. Data augmentation is the process of manipulating or modifying data in such a way that the computer detects that the manipulated data is different data, but humans can still know that the manipulated data is the same data [19]. Several studies have successfully applied data augmentation to improve the performance of the CNN method [20-25]. Therefore, providing a wide variety of imagery using data augmentation can help CNN methods in recognizing the features or characteristics of the image.

The study focused on how the use of data augmentation can help to improve accuracy in performing the recognition of handwriting image pattern using CNN methods. This recognition was done by comparing the accuracy results between baseline models and models that have been trained using data augmentation. Additionally, the recognition was also done by increasing the number of layers or networks on the CNN method through the application of LeNet-5 architecture. The CNN method was chosen because it has good accuracy for the introduction of Sundanese handwriting image patterns. Data augmentation stage is carried out after the image processing stage so that the results of image processing will be resampled to multiply the number and variation of images, the data augmentation methods used in this study are flipping (horizontal flip), rotation, and translation.

2. Research Method

The research method used in this research is to use quantitative research methods by obtaining information in the form of measurable data. The flow of this research is shown in Fig. 1.

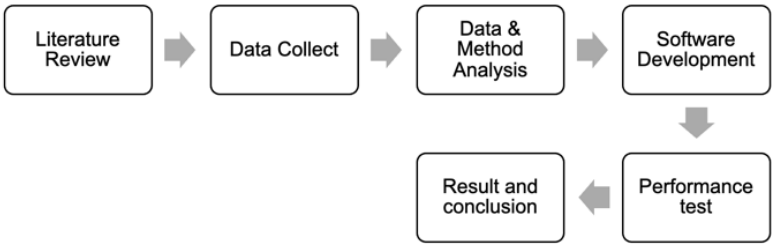

Fig. 1. Research method.

3. Analysis

The CNN network architecture used in this study refers to the LeNet-5 network architecture. The procedures of CNN architecture are shown in Figs. 2 and 3.

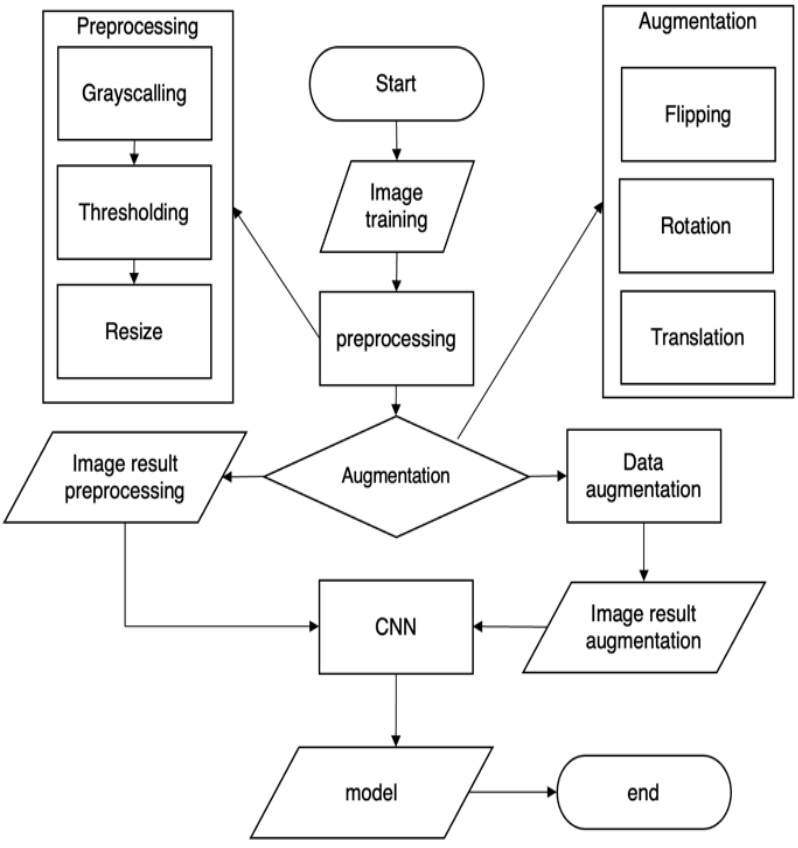

Fig. 2. Process CNN training.

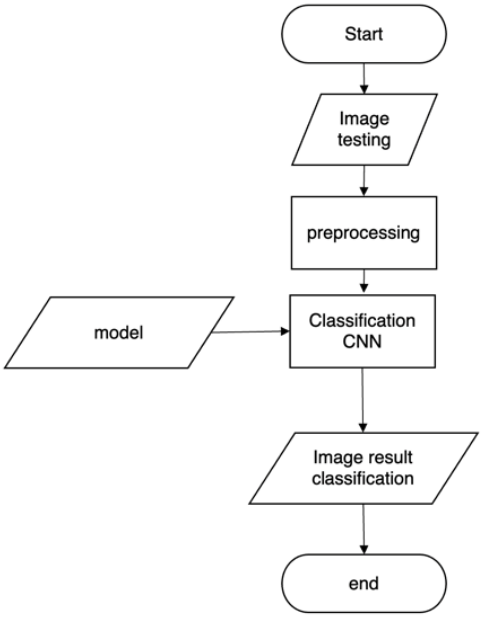

Fig. 3. Process CNN testing.

3.1. Data augmentation

The data augmentation stage is carried out after the image processing stage so that the results of image processing will be resampled to multiply the number and variation of images, the data augmentation methods used in this study are flipping (horizontal flip), rotation, and translation.

3.1.1. Flipping

The first method is flipping or flipping images, in this study horizontal flip that is to reverse the image in the direction of the x-axis using Eq. (1).

$$F_{x,y} = I_{x,(w-1-y)} \tag{1}$$

where F is the matrix of image flip, I is the matrix of original image, and w is the image width.

3.1.2. Rotation

The second method is rotation, rotating the image clockwise using Eq. (2).

$$x1 = integer\left(cos\cos\theta * \left(x2 - \left(\frac{w}{2}\right)\right) - sin\sin\theta * \left(y2 - \left(\frac{h}{2}\right)\right) + \left(\frac{w}{2}\right)\right) y1 \tag{2}$$

where *x1*, *y1* is the coordinate points on the image, *x2*, *y2* is the coordinate points on the rotation image, θ is the degree of rotation, and *w*, *h* is the image width and height.

3.1.3. Translation

The last method is translation or shifting images in the direction of the x or y axis using Eq. (3).

$$A'_{(x+a,y+b)} = A_{(x+a,y+b)} \quad (3)$$

where A is the point at the end of the image, and A' is the point at the end of the image translation. From the augmentation of the data, it produces an image with several samples as presented in Fig. 4.

Fig. 4. Data augmentation result.

3.2. CNN Architecture

The training process is carried out using a total of 1500 samples of training data. There are seven layers in LeNet-5 network architecture namely three convolutional layers, two subsampling layers, and two fully connected layers (See Fig. 5).

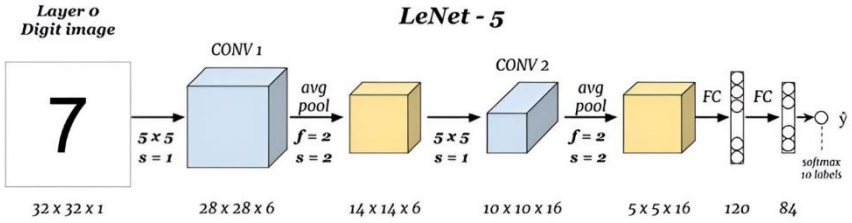

Fig. 5. LeNet-5 network architecture.

3.3. CNN training

The training process is carried out using a total of 1500 samples of training data. The following is the CNN training process.

3.3.1. Initialization

Initialization is the initial stage for assigning initial values from the weight and bias matrices that were used during the training process. Initialization of weight matrix values was done using Eq. (4) where W is the weight matrix.

$$W_{0,0,0,0} = \frac{rand(1,9)}{100} \quad (4)$$

As for the bias, it was done by giving the initial value 0.

3.3.2. Feedforward

The feedforward process in this study refers to the LeNet-5 network architecture.

(i) First convolution

The first convolution process was carried out with an input matrix in the form of an image of the previous result using Eq. (5).

$$N_{i,x,y} = \sum_m \sum_n I_{m+x,n+y} \times F0_{0,0,i,m,n} + b0_i \quad (5)$$

where *N* is the convolution matrix without activation function, *I* is the input image, *F* is the filter matrix, and *b* is the Bias filter. Then, the calculation process is carried out using the ReLU activation function as shown in Eq. (6) where $\sigma(x)$ is the ReLU activation. The equation produces a matrix of 6 matrices (neurons) with dimensions of 28×28.

$$\sigma(x) = \begin{cases} x & if\ x \geq 0 \\ 0 & other \end{cases} \quad (6)$$

(ii) First subsampling

The next process is the first subsampling process using the average polling method to reduce the dimension of the neurons generated, as shown in Eq. (7).

$$S_{x,y} = \frac{1}{fw \times fh} \sum_{u=x\times s}^{x\times s+1} \sum_{v=y\times s}^{y\times s+1} C_{u,v} \quad (7)$$

where *S* is the pooling matrix, *fw* is the width of the filter matrix dimension, *fh* is the height of the filter matrix dimension, and *C* is the convolution matrix. The equation produces 6 neurons with 14×14 dimensions.

(iii) Second convolution

The second convolution process was carried out using the same equation formula as the first convolution which produces 16 neurons with dimension of 10×10.

(iv) Second subsampling

The second subsampling process was done using the same equation formula as the first subsampling which produces 16 neurons with dimension of 5×5.

(v) Third convolution

The third convolution process was carried out using the same equation formula as the first convolution which produces 120 neurons with the dimension of 1×1. Then, it is formed into a vector flatten with dimension of 1×120.

(vi) First fully-connected

The first fully-connected process was performed against vector flattening using Eq. (8).

$$o_m = \sum_{i=0} W_{i,m} \times fi + b \quad (8)$$

where o_m is the output fully connected. The equation produces vector with dimension of 1×84.

(vii) Second fully-connected

The second fully-connected process was performed using the same equation formula as the vector of the first fully-connected result. The equation produces vector with dimension of 1×25.

(viii) Softmax

The Softmax process was carried out to normalize the previous process output. The output of this process is a vector representation of the predicted class. Output 1

means that the vector with the element index is a representation of a class that is predicted using Eq. (9) where $\hat{o}_m$ is the output softmax.

$$\hat{o}_m = \frac{e^{ym}}{\sum_{j=1} e^{ym}} \tag{9}$$

(ix) Cross entropy error function

This process was done to calculate the number of prediction errors at the time of training. If the error rate is still greater than the target error rate, then the backpropagation process is carried out. This process was done using Eq. (10) where *E* is the cross-entropy error.

$$E = -\frac{1}{n} \sum_{m=0}^{n-1} ce(m) \tag{10}$$

$$ce = \begin{cases} \ln(\hat{o}_m), & if\ (t_m = 1) \\ \ln(1 - \hat{o}_m), & other \end{cases}$$

3.4. Backpropagation

The backpropagation stage was done by calculating the derivative of the function against the cross-entropy error function. The calculation process starts from fully-connected to convolution using chain rule after which weight improvement was carried out using Stochastic Gradient Descent.

4. Results and Discussion

Performance testing

The performance testing process was applied to 375 data set by applying the testing scenario in Table 1. From the results of the tests that have been carried out, it produces output as shown in Table 2.

Table 1. Testing scenario.

No.	Testing code	Model	Epoch	Learning Rate
1	**PB01**	Baseline model	60	0.001
2	**PB02**	Baseline model	60	0.005
3	**PB03**	Baseline model	40	0.001
4	**PB04**	Baseline model	40	0.005
5	**PA01**	Baseline model + Data Augmented [Horizontal flip, Rotation=(-15°, 15°), Translation=(-2px, 2px)]	60	0.001
6	**PA02**	Baseline model + Data Augmented [Horizontal flip, Rotation=(-15°, 15°), Translation=(-2px, 2px)]	60	0.005
7	**PA03**	Baseline model + Data Augmented [Horizontal flip, Rotation=(-15°, 15°), Translation=(-2px, 2px)]	40	0.001
8	**PA04**	Baseline model + Data Augmented [Horizontal flip, Rotation=(-15°, 15°), Translation=(-2px, 2px)]	40	0.005
9	**PA05**	Baseline model + Data Augmented [Horizontal flip, Rotation=(-30°, 30°), Translation=(-4px, 4px)]	60	0.001
10	**PA06**	Baseline model + Data Augmented [Horizontal flip, Rotation=(-30°, 30°), Translation=(-4px, 4px)]	60	0.005
11	**PA07**	Baseline model + Data Augmented [Horizontal flip, Rotation=(-30°, 30°), Translation=(-4px, 4px)]	40	0.001

No.	Testing code	Model	Epoch	Learning Rate
12	**PA08**	Baseline model + Data Augmented [Horizontal flip, Rotation=(-30°, 30°), Translation=(-4px, 4px)]	40	0.005
13	**PC01**	Baseline model + Data Augmented [Horizontal flip]	60	0.001
14	**PC02**	Baseline model + Data Augmented [Horizontal flip]	60	0.005
15	**PC03**	Baseline model + Data Augmented [Horizontal flip]	40	0.001
16	**PC04**	Baseline model + Data Augmented [Horizontal flip]	40	0.005
17	**PC05**	Baseline model + Data Augmented [Rotation=(-15°,15°)]	60	0.001
18	**PC06**	Baseline model + Data Augmented [Rotation=(-15°,15°)]	60	0.005
19	**PC07**	Baseline model + Data Augmented [Rotation=(-15°,15°)]	40	0.001
20	**PC08**	Baseline model + Data Augmented [Rotation=(-15°,15°)]	40	0.005
21	**PC05**	Baseline model + Data Augmented [Translation=(-2px,2px)]	60	0.001
18	**PC06**	Baseline model + Data Augmented [Translation=(-2px,2px)]	60	0.005
19	**PC07**	Baseline model + Data Augmented [Translation=(-2px,2px)]	40	0.001
20	**PC08**	Baseline model + Data Augmented [Translation=(-2px,2px)]	40	0.005

Table 2. Results test.

No.	Testing code	Epoch	Learning Rate	Accuracy
1	**PB01**	60	0.001	0.5333
2	**PB02**	60	0.005	0.6293
3	**PB03**	40	0.001	0.5547
4	**PB04**	40	0.005	0.5467
5	**PA01**	60	0.001	0.6373
6	**PA02**	60	0.005	0.7093
7	**PA03**	40	0.001	0.5867
8	**PA04**	40	0.005	0.672
9	**PA05**	60	0.001	0.6613
10	**PA06**	60	0.005	0.7307
11	**PA07**	40	0.001	0.656
12	**PA08**	40	0.005	0.8
13	**PC01**	60	0.001	0.5600
14	**PC02**	60	0.005	0.584
15	**PC03**	40	0.001	0.5706
16	**PC04**	40	0.005	0.5920
17	**PC05**	60	0.001	0.6746
18	**PC06**	60	0.005	0.6586
19	**PC07**	40	0.001	0.6453
20	**PC08**	40	0.005	0.6293
21	**PC09**	60	0.001	0.6906
22	**PC10**	60	0.005	0.6693
23	**PC11**	40	0.001	0.6053
24	**PC12**	40	0.005	0.6613

From the results of the tests that have been carried out, the baseline model with the addition of data augmentation in the dataset has a significant effect. The highest accuracy is obtained at a value of 0.8 in the 12th test (PA08) with an epoch value

of 40 and a learning rate of 0.005. Whereas for the baseline model without data augmentation, the highest accuracy value was 0.6293 in the second test (PB02) with an epoch value of 60 and a learning rate of 0.005. The difference between the highest accuracy value of the baseline model with data augmentation and without data augmentation is 0.1707. The difference is significant enough to prove that adding data augmentation process improves the accuracy value.

The use of data augmentation has a significant effect on model performance. The use of data augmentation with a horizontal flip configuration, rotation with a random range between -30° to 30°, and translation with a random range between -4px to 4px are proven to be better than just using a horizontal configuration flip, rotation with a random range between -15° to 15°, and translation with a random range between -2px to 2px. This is sufficient to prove that increasingly varied datasets can increase the accuracy value.

The data augmentation technique also affects performance. In performance testing, the horizontal flip data augmentation technique does not have much effect on improving model performance when compared to the baseline model. Meanwhile, the use of data rotation and translation augmentation techniques has the greatest influence on improving model performance.

Differences in the use of epoch and learning rate parameters also affect model performance, using higher epoch values and learning rates tend to get better results. However, in several testing experiments on the baseline model there was an anomaly in the third experiment with a lower epoch value which produces a higher accuracy value than a higher epoch of the first experiment.

5. Conclusion

Based on the results of research that has been done on data augmentation for the recognition of handwriting image patterns using the CNN method, it can be concluded that the use of data augmentation to increase the number and variety of data samples has a significant effect on the accuracy value of 0.1707 or 17.07%. The best accuracy obtained is 0.8 or 80% using a baseline model with data augmentation. These findings yielded good results because the system is able to perform image classification quite well.

References

1. Ramadhan, T.I.; Ramadhan, N.G.; and Supriatman, A. (2022). Implementation of neural machine translation for English-Sundanese language using long short-term memory (LSTM). *Building of Informatics, Technology and Science (BITS)*, 4(3), 1438-1446.
2. Ibrahim, F.S.; Elsamani, E.; and Siddig, S. (2021). Sentiment detection on Sundanese political tweets using deep learning approach. *International Journal of New Technology and Research*, 7(2), 30-34.
3. Permanasari, Y.; Ruchjana, B.N.; Hadi, S.; and Rejito, J. (2022). Innovative region convolutional neural network algorithm for object identification. *Journal of Open Innovation: Technology, Market, and Complexity*, 8(4), 182-191.

4. Gunarto, H. (2019). Apps-based machine translation on smart media devices-a review. *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*, 13(1), 95-104.
5. Shorten, C.; and Khoshgoftaar, T.M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1), 1-48.
6. Balaha, H.M.; Ali, H.A.; and Badawy, M. (2021). Automatic recognition of handwritten Arabic characters: A comprehensive review. *Neural Computing and Applications*, 33(2), 3011-3034.
7. Shorten, C.; and Khoshgoftaar, T.M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1), 1-48.
8. Atmaja, B.; Akagi, M.; and Elbarougy, R. (2020). Dimensional speech emotion recognition from acoustic and text features using recurrent neural networks. *International Journal of Informatics, Information System and Computer Engineering (INJIISCOM)*, 1(1), 91-102.
9. Devi, C.A.; Abdul Jabbar, F.; Varshini, S.K.; Rithanya, K.; Miruthubashini, M.; and Naveena, K.S. (2021). Risks of chronic kidney disease prediction using various data mining algorithms. *International Journal of Informatics, Information System and Computer Engineering (INJIISCOM)*, 2(2), 53-65.
10. Iqbal Qatrunada, M.; Fadhlih Sephia Harasta, M.; and Tosofu, I. (2022). Utilization of augmented reality technology as an interactive learning media. *International Journal of Research and Applied Technology (INJURATECH)*, 2(1), 188-195.
11. Suryadjaja, P.; Hutagalung, M.; and Sutarto, H. (2020). Modeling traffic flows with fluid flow model. *International Journal of Informatics, Information System and Computer Engineering (INJIISCOM)*, 1(1), 1-12.
12. Goonjur, M.; Sumitra, I.; and Supatmi, S. (2020). Enhanced the weighted centroid localization algorithm based on received strength signal in indoor wireless sensor network. *International Journal of Informatics, Information System and Computer Engineering (INJIISCOM)*, 1(1), 13-22.
13. Pangaribuan, I.; Rahman, A.; and Mauluddin, S. (2020). Computer and network equipment management system (CNEMAS) application measurement. *International Journal of Informatics, Information System and Computer Engineering (INJIISCOM)*, 1(1), 23-34.
14. Shen, L.; Chen, B.; Wei, J.; Xu, H.; Tang, S.K.; and Mirri, S. (2023). The challenges of recognizing offline handwritten Chinese: A technical review. *Applied Sciences*, 13(6), 3500-3529.
15. Kaur, S.; and Kulkarni, N. (2021). Emotion recognition - A review. *International Journal of Applied Engineering Research*, 16(2), 103-110.
16. Kiranyaz, S.; Avci, O.; Abdeljaber, O.; Ince, T.; Gabbouj, M.; and Inman, D.J. (2021). 1D convolutional neural networks and applications: A survey. *Mechanical Systems and Signal Processing*, 151(1), 107398-107419.
17. Lindsay, G.W. (2021). Convolutional neural networks as a model of the visual system: Past, present, and future. *Journal of Cognitive Neuroscience*, 33(10), 2017-2031.
18. Dhillon, A.; and Verma, G.K. (2020). Convolutional neural network: A review of models, methodologies and applications to object detection. *Progress in Artificial Intelligence*, 9(2), 85-112.

19. Kattenborn, T.; Leitloff, J.; Schiefer, F.; and Hinz, S. (2021). Review on convolutional neural networks (CNN) in vegetation remote sensing. *ISPRS Journal of Photogrammetry and Remote Sensing*, 173(1), 24-49.
20. Naranjo-Torres, J.; Mora, M.; Hernández-García, R.; Barrientos, R.J.; Fredes, C.; and Valenzuela, A. (2020). A review of convolutional neural network applied to fruit image processing. *Applied Sciences*, 10(10), 3443-3473.
21. Girsang, N.D. (2021). Literature study of convolutional neural network algorithm for batik classification. *Brilliance: Research of Artificial Intelligence*, 1(1), 1-7.
22. Dhaka, V.S.; Meena, S.V.; Rani, G.; Sinwar, D.; Ijaz, M.F.; and Woźniak, M. (2021). A survey of deep convolutional neural networks applied for prediction of plant leaf diseases. *Sensors*, 21(14), 4749-4782.
23. Ali, R.; Chuah, J.H.; Talip, M.S.A.; Mokhtar, N.; and Shoaib, M.A. (2022). Structural crack detection using deep convolutional neural networks. *Automation in Construction*, 133(1), 103989-104010.
24. Tulbure, A.A.; Tulbure, A.A.; and Dulf, E.H. (2022). A review on modern defect detection models using DCNNs–deep convolutional neural networks. *Journal of Advanced Research*, 35(1), 33-48.
25. Sultana, F.; Sufian, A.; and Dutta, P. (2020). A review of object detection models based on convolutional neural network. *Intelligent Computing: Image Processing Based Applications*, 1157(1), 1-16.