Journal of Engineering Science and Technology
Vol. 21, No. 2 (2026) 597 - 606

DENSENET-121 FOR HANDWRITTEN SUNDANESE SCRIPT IMAGE PATTERN RECOGNITION

NELLY INDRIANI WIDIASTUTI*, NURI CHANDRA

Faculty of Engineering and Computer Science,
Universitas Komputer Indonesia, Bandung, Indonesia
*Corresponding Author: nelly.indriani@email.unikom.ac.id

Abstract

Sundanese script is a type of script that is increasingly rarely used, and therefore, many people are no longer familiar with it. The difficulty in recognising Sundanese script arises due to variations in handwriting forms. The aim of this study is to measure the performance of densenet-121 in recognizing Sundanese handwriting, both in the form of letters and combinations between basic letters and *rarangkén* (XXX). The approach adopted in this study involves using the DenseNet-121 model to classify 17,280 handwritten Sundanese script images into 144 classes of script images, after applying grayscale conversion, thresholding, segmentation, and resizing. This model was evaluated based on accuracy, precision, recall, and F1-score. The test results showed that the best model utilised the AdamW optimiser, a learning rate of 0.001, and 10 epochs, which produced an accuracy of 79.5%, a precision of 63.2%, a recall of 65.3%, and an F1-score of 62.0%. The evaluation results showed fairly good accuracy. However, there was a caveat: the model suffers from bias. This was due to the high diversity of the data. Unique features in the data need to be generalized. Compared to previous studies, the larger dataset in this study provided broader coverage, but also added complexity that needs to be considered to improve model performance in the future.

Keywords: Deep learning, DenseNet-121, Handwriting, Pattern recognition, Sundanese script.

1. Introduction

Sundanese script is a form of writing used by the Sundanese people since around the 5th century AD [1]. This script consists of swara (vowels), ngalagena (consonants), special characters, rarangkén (vocalisation marks), pairs, and numbers. The ngalagena script is a symbol that represents consonant sounds and is always accompanied by the vowel sound "a". This script consists of 18 symbols that reflect sounds in Sundanese, plus five symbols from adopted sounds and two additional symbols [2]. Sundanese script rarangkén are symbols that show how to pronounce vowels in the Sundanese script. Rarangkén are very important because they help clarify the meaning and intonation of words, making them easier to read [2]. The Sundanese script has a distinctive and unique character form, with a variety of basic characters and additional symbols. Variations in individual writing styles also add to the difficulty, as each person can have a different writing style [3, 4].

DenseNet-121 is a Convolutional Neural Networks (CNN) architecture that has proven effective in various pattern recognition tasks, including handwriting. Several previous studies used the DenseNet-121 architecture. The first study presents a comparison of the performance of various CNN architectures in detecting Arabic handwritten images [4]. They used four architectures and showed that DenseNet-121 achieved an accuracy of 98.42% for training data and 97.63% for testing data [4]. Other research regarding the use of CNN in the classification of handwritten Bangla letters, which showed that DenseNet obtained the highest accuracy of 98.31% [5]. Several other studies utilised the DenseNet-121 architecture to classify weather imagery with 95.9% accuracy [6].

Previous research applied Convolutional Neural Networks to classify Sundanese handwriting [7, 8]. They used a diverse image dataset of Sundanese scripts ngalagena and swara, which achieved an accuracy of 85.5% [7]. Another study using CNN detected 10 classes, each with 5 images [8]. Research on Sundanese script recognition has so far been limited to character detection; in some words, the ngalagena script is often combined with rarangkén to form different sounds [9]. Therefore, this study contributes to classifying Sundanese handwriting patterns in the 'ngalagena' and 'rarangken' cases, a problem previously unaddressed. We used DenseNet-121, which can access features at all layers and reuse its learning outcomes.

2. Literature Review

2.1. Sundanese script

Research focused on Sundanese script objects tested a CNN to recognise the letters "ngalagena" and "swara" with 2,325 training data, and various methods of collecting test data produced an accuracy of 85,5% [8]. Another study employed CNN to test 944 training and test data for only the "Swara" category, resulting in 92% accuracy for data testing [10]. In addition, a different study implemented LVQ and MDF on 300 data, achieving an accuracy of 78.67% [11]. Therefore, previous research used ANN and Canny edge detection to test 91 data points, showing an accuracy 76.19% [12]. Another study examined the influence of data augmentation. They obtained a 17% improvement from 375 test data [13]. Apart from image data, Sundanese language was also developed from text data in the form of a translation machine [14].

2.2. DenseNet-121

DenseNet is included in the CNN family, which has been widely used for object detection, including letter recognition, image captioning [15], or MRI image detection. In this section, several studies have used DenseNet-121 and shown quite good results. Although not Sundanese script, previous research tested several CNN architectures and showed quite good results [16]. In addition, the other study detected Bangla script using several CNN architectures, including DenseNet. The results obtained showed that DenseNet was superior [6]. A study utilized weather image data and demonstrated that DenseNet achieved an accuracy of 85% [7]. Hangul script was the object of research using DCNN [17].

3. Method

This research was conducted in several stages. The first stage was collecting handwritten images of Sundanese script. After that, data preprocessing was carried out, which included several steps, namely conversion to grayscale images (grayscaling), thresholding, contour-based segmentation to separate the ngalagena and rarangkén scripts and resizing. Next, the dataset was divided into 80% for training and 20% for validation. The classification model used was DenseNet-121, which was trained on the dataset to recognise handwritten patterns of Sundanese script. The final stage evaluated the model's performance using accuracy, precision, recall, and F1-score metrics to assess system performance in classifying Sundanese script.

3.1. Dataset

The dataset is an important element because it is used to validate the performance of the detection method. The dataset must be able to measure how robust the proposed method is [18]. The images used for training and validation image samples were taken from the Kaggle, containing 18 letters of the Sundanese script. For the test data, there were 60 respondents who each wrote 18 letters of the Sundanese script that had been combined with seven sequences (panghulu (i), panyuku (u), panéléng (é), pamepet (e), paneuleung (eu), panolong (o), pamaéh) as shown in Table 1.

Table 1. Proportion dataset.

Data Type	Sub Total	Total
Data Ngalagena (18 classes)	120x18	2160
Data Combination Ngalagena and Rarangken (126 classes)	120x126	15120
Total		17280

Figure 1 shows how the shape of the Sundanese script, accompanied by rarangkén, influences the sound. This is important to recognise because naturally written Sundanese script always includes rarangkén.

Fig. 1. Sundanese character with rarangkén.

Meanwhile, 967 test images were separated from the start. The data consisted of handwritten Sundanese script words. Each letter was written with a black pen on white HVS paper. In this study, the 17,280 data were divided into 80% for training data and 20% for validation. For testing, 967 test images were prepared, in the form of words written in Sundanese script by several respondents. All images were written with black ink on white paper. all images are in RGB format, with a size of 355x355 pixels for ngalagena letters, while for combinations with rarangkén, the size varied.

3.2. Preprocessing

To reduce computational complexity, the dataset was pre-processed in the form of grayscale. The image was rendered to have colour intensities in the range of black, grey, and white, with a colour depth of eight bits, allowing for 256 combinations of grayscale. By converting the colour format to grayscale, the colour range could be narrowed. After grayscaling, the image converted an image into binary form. Then, the segmentation process isolated each character, enabling the system to recognise them. This process begins by detecting contours or edges in the thresholded image. Contours that are close together, such as basic letters (ngalagena) and diacritical marks (rarangkén) in Sundanese script, were grouped into a character unit. After the contours are grouped, each group was cropped from the original image so that each piece represented a single character. The last step was to resize the image. Images with dimensions smaller than 224 x 224 were enlarged to 224 x 224. Conversely, if the image had dimensions larger than 224 x 224, it was reduced. The resizing process uses the Nearest Neighbour Interpolation method. This method works by selecting the closest pixel value by rounding the coordinates of the point to be changed.

3.3. DenseNet-121

DenseNet-121 is a dense CNNwith 121 layers, consisting of four interconnected dense blocks. This architecture begins with convolutional and pooling layers, ending with a classification layer that uses the softmax function to calculate the probability of each class [19]. The architectural details are shown in Fig. 2.

(i) Zero Padding: In this study, padding = 'same' was applied to prevent information loss at the edges of the image during the convolution operation.
(ii) Convolution 7x7 strides 2: Convolution extracted features from input images by applying a set of randomly initialised filters. Each filter produced a feature map that captured the patterns, edges, and textures of the image [20]. The values of the pixels in the table were multiplied by the 7x7 kernel.
(iii) Batch Normalisation: This technique equalised the distribution of input values that were are constantly changing due to parameter changes in the previous layer during the training process [21]. The batch normalisation operation was applied using an 8x8 matrix generated from convolution.
(iv) ReLU Activation: ReLU function worked by converting all negative values to 0, while keeping values greater than 0 unchanged. This function helped the model learn faster and overcomes the problem when the gradient (change in value) becomes very small [22].
(v) Max Pooling: The goal of this process was to lower data dimensionality, thus decreasing the number of parameters processed by the network while preserving the essential features of the original image. Max Pooling was performed using a 3x3 stride size of 2 [23].

(vi) Average Pooling: the goal was to reduce the size of feature representations. By averaging the values of features within a given window or kernel (e.g., 2x2), average pooling simplified information while preserving important features [23].
(vii) Global Average Pooling: global average pooling calculates the average of the entire feature area, producing one value for each channel [24].
(viii) Softmax Activation: This function is especially beneficial when the model needs to select from multiple classes, as it indicates the model's confidence for each class. Softmax calculates the probability distribution of a vector of real numbers, producing an output between 0 and 1 [25].
(ix) Cross-Entropy: Its function is to measure the relative entropy between two probability distributions on the same data. In this study, categorical cross-entropy was used.

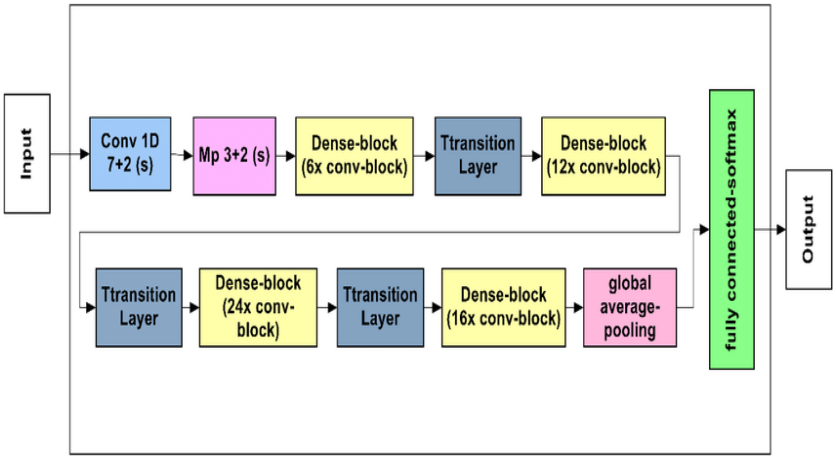

Fig. 2. DenseNet-121 architecture [19].

4. Results and Discussion

4.1. Results

The evaluation involved several parameter settings—epochs of 5 and 10, learning rates of 0.001 and 0.0001, and both Adam and Adam W optimisers —as shown in Table 2. Furthermore, Test 5 incorporated rotation and translation data augmentation during training [26].

Table 2. Training hyperparameters and experimental scenarios used in model evaluation.

Hyperparameter	Value	Scenario	Data Augmentation
Epoch	5, 10	Test 1-4	No
Learning Rate	0.001, 0.0001	Test 5	Yes
Optimizer	Adam, AdamW		

Epochs 5 and 10 were chosen to achieve a balance between training time and model performance. In Fig. 3, training accuracy remained stable above 98%, but validation accuracy fluctuated with several sharp drops. The loss graph also shows

a spike during validation, indicating the model is struggling to generalise. Results obtained using the test data only reached 69%. Therefore, 10 epochs were chosen to prevent overfitting and maintain a balance between accuracy and generalisation.

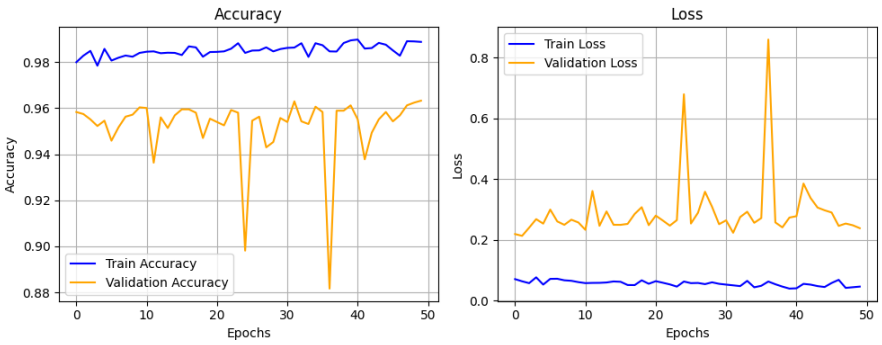

Fig. 3. Plot accuracy and loss epoch 50.

4.1.1. Best results

The fourth experiment achieved the best result at epoch 10 at learning rates of 0.001 and 0.0001 using the AdamW optimiser. Figure 4 shows an increase in model accuracy. However, validation accuracy fluctuated, indicating possible overfitting. The loss graph shows that the training loss continued to decrease steadily, contrary to the validation loss, which experienced a spike, thereby indicating the model was overfitting to the training data.

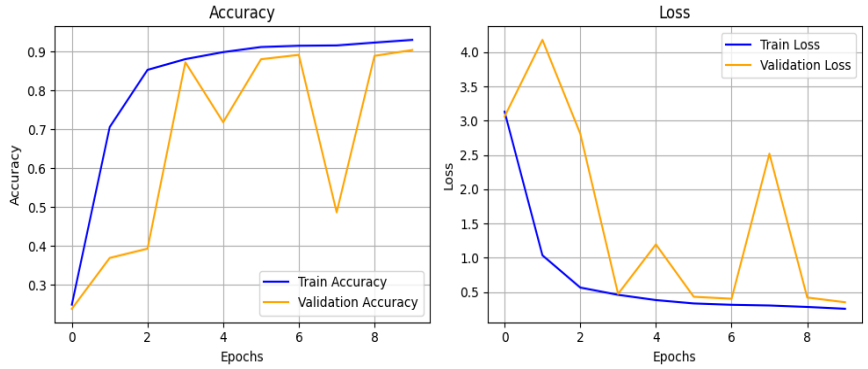

Fig. 4. Plot accuracy and loss for learning rate 0.001.

Figure 5 shows that the accuracy increased more steadily than in the previous test, indicating better model generalization. Furthermore, the loss graph shows that both training and validation losses decreased consistently without spikes, showing a more stable model. In the fourth classification test report with a learning rate of 0.001, the character with the highest F1-score is the character “du”. In the fourth classification test report with a learning rate of 0.0001, the character with the highest F1-score is the character “beu”.

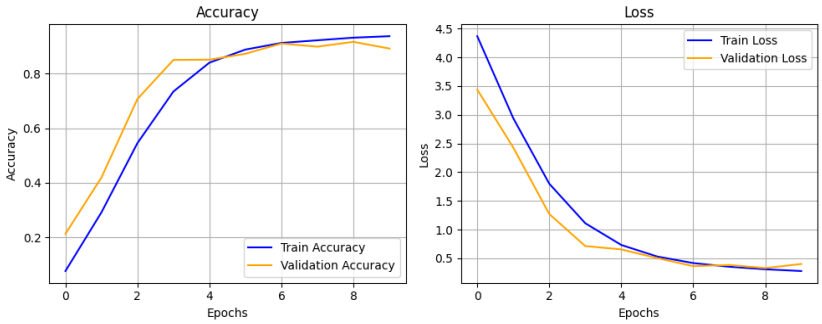

Fig. 5. Plot accuracy and loss for learning rate 0.0001.

4.1.2. Augmentation data testing

The fifth test used epochs 5 and 10 at a learning rate of 0.001 and the AdamW optimiser. Figure 6 shows epoch 5 for both training and validation data. Initially, the model learned well, but in the final epoch, validation accuracy began to decline, while validation loss increased. This indicates that the model was experiencing overfitting.

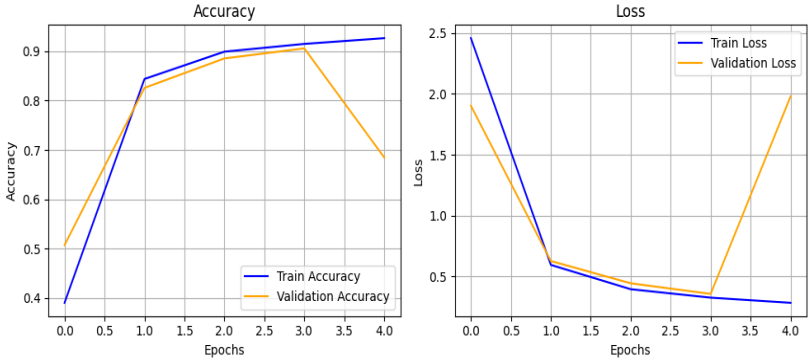

Fig. 6. Plot accuracy and loss for epoch 5.

Figure 7 shows epoch 10, which shows that overfitting became more apparent, training accuracy continued to rise, but validation accuracy fluctuated and eventually dropped. Training loss decreased, but validation loss spiked towards the end of training.

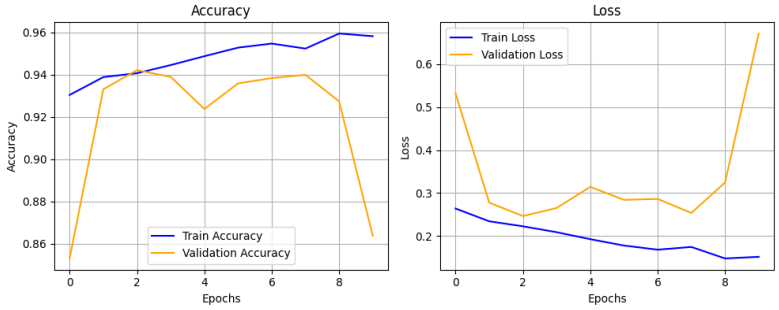

Fig. 7. Plot accuracy and loss for epoch 10.

In the fifth test classification report with epoch 5, the character with the highest F1-score was character “d”. In the fifth test classification report with epoch 10, the character with the highest f1-score was the character “beu”. Data augmentation did not provide a significant improvement of results compared to the fourth test.

4.2. Discussion

Following the completion of tests 1 through 5, test 1 revealed that the model struggled to comprehend the validation pattern, while tests 2 and 3 exhibited volatility in validation values. This suggests a propensity towards overfitting. The optimal results are presented in Table 3, the best model emerged from the fourth test, which utilised the AdamW optimiser, a learning rate of 0.001, and 10 epochs. This model achieved an accuracy of 79.5%, as well as precision, recall, and f1-score values that indicate a good balance in model performance. High accuracy demonstrates that the model classifies data with greater precision, while superior precision and recall imply enhanced capability in recognising various classes compared to other models. The loss produced by this model is also the lowest, which indicates that the model successfully minimises prediction errors.

Table 3. Results summary.

Test	Data Augmentation	Accuracy	Precision	Recall	F1-score	Learning rate	Epoch
4	No	79.5%	63.2%	65.3%	62.0%	0.001	10
5	Yes	77.0%	63.0%	60.8%	59.2%	0.001	10

On the other hand, testing with training data augmented with rotation and translation did not result in improved accuracy. Although the model trained for 10 epochs achieved higher accuracy than the one with five epochs, its performance remained below that of the best model without augmentation. Furthermore, the recall at epoch 5 was lower, and the loss at epoch 10 increased to 1.4267, indicating that the model did not learn better. The excessive rotation angle or translation distance likely distorted the character's shape, making it difficult for the model to recognise.

Several previous studies related to Sundanese script handwriting recognition have been conducted using various methods and different datasets. Previous research utilised a CNN with a Sundanese script ngalagena and swara dataset comprising 31 classes with 2,325 training data and 62 test data for handwriting images captured via cell phone camera, achieved an accuracy of 87.41% [8]. In addition, a study utilised a Sundanese script ngalagena and swara dataset of seven classes with 944 data points, achieving the best accuracy of 92% [10].

These differences in results indicate that factors such as dataset type, data processing techniques, and model architecture greatly influence the level of accuracy obtained in Sundanese script handwriting pattern recognition. An advantage of this study is its large dataset, comprising 17,280 training samples, 967 test samples, and 144 classes, making it larger than those used in previous studies.

5. Conclusion

From the test results conducted on the use of Densenet-121 for image pattern recognition of handwritten Sundanese script, the best model was obtained with the AdamW optimiser, with a learning rate 0.001, and 10 epochs, which produced the

highest accuracy of 79.5%. Although the model was able to recognise various classes quite well, weaknesses still existed due to the large number of classes and the distribution of test data that did not cover all classes in the training data, thus affecting overall performance. Compared to previous studies, the larger dataset in this study provided broader coverage, but also added complexity that will need to be considered to improve model performance in the future. Improvements can be made by searching for the best parameters or paying more attention to the data distribution in each class so that the model works better.

References

1. Sarrahdiba, T.U.Y. (2014). Perancangan Media Pembelajaran Aksara Sunda Untuk Siswa Sekolah Dasar (Studi Kasus SDN Sukasenang). *Jurnal Sketsa*, 1(1), 24-31.
2. Lestari, A.; Nurizki, A.; and Hanifah, H.G. (2023). Analisis perbandingan fonem bahasa Sunda dan bahasa Indonesia. *Sintaksis: Publikasi Para Ahli Bahasa dan Sastra Inggris*, 1(6), 62-71.
3. Widoretno, S.; Sarosa, M.; and Muslim, M.A. (2013). Implementasi pengenalan karakter seseorang berdasarkan pola tulisan tangan. *Jurnal EECCIS (Electrics, Electronics, Communications, Controls, Informatics, Systems)*, 7(1), 97-102.
4. Fadhilla, M.; Saf, M.R.I.A.; and Sahid, D.S.S. (2017). Pengenalan kepribadian seseorang berdasarkan pola tulisan tangan menggunakan jaringan saraf tiruan. *Jurnal Nasional Teknik Elektro dan Teknologi Informasi*, 6(3), 365-373.
5. Boraik, O.A.; Ravikumar, M.; and Chola, C. (2022). Arabic handwritten character classification and recognition approach using and transfer. *The Seybold Report*, 17(11), 2087-2107.
6. Alom, M.Z.; Sidike, P.; Hasan, M.; Taha, T.M.; and Asari, V.K. (2018). Handwritten bangla character recognition using the state-of-the-art deep convolutional neural networks. *Computational Intelligence and Neuroscience*, 2018(1), 1-13.
7. Albelwi, S.A. (2022). Deep architecture based on DenseNet-121 model for weather image recognition. *International Journal of Advanced Computer Science and Applications*, 13(10), 559-565.
8. Kirana, A.; Hikmayanti, H.; and Indra, J. (2020). Pengenalan pola aksara sunda dengan metode convolutional neural network. *Scientific Student Journal for Information, Technology and Science*, 1(2), 95-100.
9. Purnama, A.; Bahri, S.; Gunawan, G.; Hidayatulloh, T.; and Suhada, S. (2022). Implementation of deep learning for handwriting imagery of sundanese script using convolutional neural network algorithm (CNN). *ILKOM Jurnal Ilmiah*, 14(1), 10-16.
10. Rahmawati, S.N.; Hidayat, E.W.; and Mubarok, H. (2021). Implementasi deep learning pada pengenalan aksara sunda menggunakan metode convolutional neural network. *INSERT: Information System and Emerging Technology Journal*, 2(1), 46-58.
11. Riansyah, R.R.; Nurhasanah, Y.I.; and Dewi, I.A. (2017). Sistem pengenalan aksara sunda menggunakan metode modified direction feature dan learning vector quantization. *Jurnal Teknik Informatika and Sistem Informasi*, 3(1), 17-30.

12. Amalia, N.; Hidayat, E.W.; Aldya, A.P.; and No, U.S.J.S. (2020). Pengenalan aksara sunda menggunakan metode jaringan saraf tiruan backpropagation dan deteksi tepi canny. *CESS (Journal of Computer Engineering, System and Science)*, 5(1), 19-26.
13. Maliki, I.; and Prayoga, A.S. (2023). Implementation of convolutional neural network for sundanese script handwriting recognition with data augmentation. *Journal of Engineering Science and Technology*, 18(2), 1113-1123.
14. 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*, 4(3), 1438-1446.
15. Nursikuwagus, A.; Munir, R.; and Khodra, M.L. (2020). Image captioning menurut scientific revolution kuhn dan popper. *Jurnal Manajemen Informatika (JAMIKA)*, 10(2), 110-121.
16. Masruroh, S.U.; Syahid, M.F.; Munthaha, F.; Muharram, A.T.; and Putri, R.A. (2023). Deep convolutional neural networks transfer learning comparison on arabic handwriting recognition system. *JOIV: International Journal on Informatics Visualization*, 7(2), 330-337.
17. Kim, I.J.; and Xie, X. (2015). Handwritten hangul recognition using deep convolutional neural networks. *International Journal on Document Analysis and Recognition (IJDAR)*, 18(1), 1-13.
18. Rainarli, E. (2021). A decade: Review of scene text detection methods. *Computer Science Review*, 42, 100434-100451.
19. Tareq, I.; Elbagoury, B.M.; El-Regaily, S.; and El-Horbaty, E.S.M. (2022). Analysis of ton-IOT, unw-nb15, and edge-IIOT datasets using dl in cybersecurity for IOT. *Applied Sciences*, 12(19), 9572-9585.
20. Žbontar, J.; and LeCun, Y. (2016). Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research*, 17(65), 1-32.
21. Nurhaida, I.; Ayumi, V.; Fitrianah, D.; Zen, R.A.; Noprisson, H.; and Wei, H. (2020). Implementation of deep neural networks (DNN) with batch normalization for batik pattern recognition. *International Journal of Electrical and Computer Engineering*, 10(2), 2045-2053.
22. Lin, G.; and Shen, W. (2018). Research on convolutional neural network based on improved relu piecewise activation function. *Procedia Computer Science*, 131, 977-984.
23. Zafar, A.; Aamir, M.; Mohd Nawi, N.; Arshad, A.; Riaz, S.; Alruban, A.; and Almotairi, S. (2022). A comparison of pooling methods for convolutional neural networks. *Applied Sciences*, 12(17), 8643-8658.
24. Lu, X.; Moffat, A.; and Culpepper, J.S. (2016). The effect of pooling and evaluation depth on IR metrics. *Information Retrieval Journal*, 19(4), 416-445.
25. Dubey, S.R.; Singh, S.K.; and Chaudhuri, B.B. (2022). Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, 503, 92-108.
26. Shorten, C.; and Khoshgoftaar, T.M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1), 1-48.