Building of Informatics, Technology and Science (BITS)
Volume 7, No 2, September 2025 Page: 984−992
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i2.7625

Implementation of an Artificial Neural Network in the Classification of Handwritten Javanese Script Images

Zainuri Rohim, Mohammad Nasucha*

Department of Informatics, and Center for Urban Studies, Universitas Pembangunan Jaya, Tangerang Selatan, Indonesia
Email: [1]zainuri.rohim@student.upj.ac.id, [2,*]mohammad.nasucha@upj.ac.id
Correspondence Author Email: mohammad.nasucha@upj.ac.id
Submitted: **18/06/2025**; Accepted: **01/09/2025**; Published**: 02/09/2025**

Abstract−Javanese script is an Indonesian cultural heritage rich in historical, aesthetic, and spiritual values, but it is now becoming marginalized. To reintroduce its use, this research develops a Javanese script recognition application based on an Artificial Neural Network (ANN). In this study, the Javanese script was divided into 120 classes (ha, hi, hu, he, hee, ho, up to nga, ngi, ngu, nge, ngee, ngo). Each class was represented by 40 sample images of the script handwritten by 40 different respondents, resulting in 4800 samples. The research began with preprocessing, which included adding padding to the top, bottom, left, and right sides of the script; downsizing the image to a 33x33 resolution by applying average pooling; image segmentation to separate the script characters from the background; converting the color image to grayscale; and converting the grayscale image to a binary image with the help of thresholding. A number of images that had undergone preprocessing were then structured into a ready-to-use dataset of 4800 samples. This dataset was then divided with an 80:20 ratio, where 80% of the data was used to train the model and 20% was used to validate the model. An evaluation was conducted to measure the model's accuracy. Subsequently, the application was developed using PySide6 as the desktop interface. After the application development, the researchers provided an additional 600 images, where each class was represented by 5 samples, for real-world application testing. The evaluation results showed that the model achieved a validation accuracy of 70.21%. Meanwhile, testing with the application using the additional test images showed an accuracy of 73.83%.

Keywords: Javanese Script; Character Recognition; Artificial Neural Network; Image Segmentation; Pyside6

1. INTRODUCTION

A script is a system of writing language that is visually expressed on various media such as paper, stone, wood, cloth, and so on. Throughout the world, there are various types of scripts, including the Latin script, Arabic script, and others. Indonesia, as one of the largest archipelagic nations, is rich in cultural diversity, including in terms of scripts. In Indonesia itself, there are various scripts such as the Makassar script, Batak script, Javanese script, Sasak script, and others [1]. This research will focus on the Javanese script.

Javanese script is one of Indonesia's cultural assets, rich with significant historical value, beauty, and spiritual dimensions. As part of the cultural identity of the Javanese people, this script has long been used in various ancient manuscripts, carving arts, to traditional writing [2]. However, with the rapid development of digital technology, the use of Javanese script in daily life has experienced a significant decline [3]. The limitation of interactive learning media and the lack of integration of Javanese script in modern technology have also become factors causing the reduced interest of the younger generation in this script. To support its preservation, a tool or software that has the ability to automatically recognize Javanese script characters is needed.

Various studies have been conducted to answer these challenges, especially in the effort to automatically recognize Javanese script using an artificial intelligence approach. The majority of studies use Convolutional Neural Network (CNN) due to its ability to automatically and effectively extract visual features from images. For instance, research by Diyasa applied CNN for the classification of Javanese script characters and showed good performance in visual pattern recognition [4]. Another CNN approach was also used by Septhian et al., who implemented the MobileNetV2 architecture, known for being lighter yet still accurate for the classification of Javanese script writing [5]. Besides CNN, classical learning-based approaches are also widely used, such as Learning Vector Quantization (LVQ). S. Hamzah used the LVQ method based on morphological features to recognize handwritten Javanese script with a distance measurement approach using Euclidean Distance [6]. Another approach based on K-Nearest Neighbor (KNN) was also implemented by A. Susanto, who utilized Local Binary Pattern (LBP) as a feature extraction method. The research results showed that KNN was able to effectively classify Javanese script characters based on the local texture patterns of the image [7]. On the other hand, modern object detection approaches have also begun to be applied, such as in the research by Faizin who used YOLO (You Only Look Once) for the detection of Javanese script characters in ancient manuscripts, then performed transliteration using an LSTM (Long Short-Term Memory) based model [8]. Although the results achieved are quite promising, most of the previous research still has limitations, particularly in the size of the dataset and the scope of characters used. Generally, these studies only focused on the recognition of single characters with a limited number of characters, around 20 scripts only. In this research, a recognition system for nglegana script equipped with sandhangan swara is developed, covering a total of 120 characters, each having various handwriting variations.

Besides targeting the recognition of more complex handwriting, this research also aims to provide a larger and more varied dataset that can be utilized in further studies in the field of Javanese script preservation. Therefore, an Artificial Neural Network (ANN) approach is used, specifically designed to recognize diverse forms of the script. Not only for cultural preservation, this system also has the potential to be further developed into an application capable of

Building of Informatics, Technology and Science (BITS)
Volume 7, No 2, September 2025 Page: 984−992
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i2.7625

recognizing whole sentences in Javanese script and translating them into the Indonesian language, thereby expanding its benefits in the context of education and cultural digitalization.

2. RESEARCH METHODOLOGY

An Artificial Neural Network (ANN) is an artificial intelligence method inspired by the way the human brain processes information and recognizes patterns [9][10]. An Artificial Neural Network (ANN) consists of processing units called neurons [11]. Each neuron receives a number of inputs in the form of a vector x, where each is multiplied by a weight w. The result of this multiplication is summed and then a bias value b is added. This operation produces a total value called the initial linear activation or pre-activation, z. The value z is then processed by an activation function f to produce the neuron's final output, which is mathematically formulated as:

$$z = \sum_{i=1}^{n} w_i x_i + b \quad ()$$

$$y = f(z)$$

The activation function in an ANN plays a crucial role in introducing non-linear properties into the network, enabling the ANN to learn complex relationships between the input and target output [12]. In this research, the sigmoid activation function is used. The sigmoid function was chosen because it produces a continuous output between 0 and 1, which is suitable for probability-based classification approaches [13][14].

2.1 Research Stages

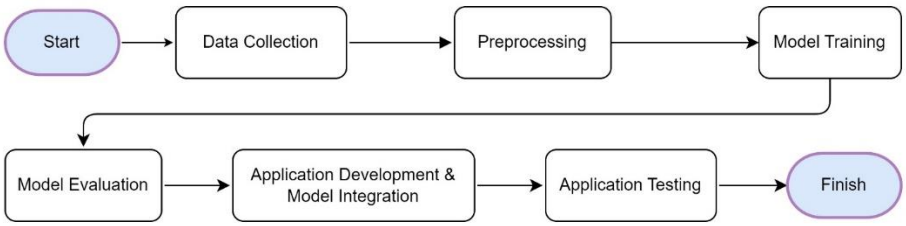

Figure 1. Research Stages

This research follows a series of systematic stages presented in the form of a flowchart. Each stage has a crucial role in ensuring the success of the Javanese script recognition system being built. The sequence of stages in this research includes: data collection, preprocessing, model training, model evaluation, application development and model integration, and application testing, as visualized in Figure 1 above.

2.1.1 Data Collection

The initial step in this research began with the data collection process. The data collected consists of handwritten Javanese script, specifically the nglegana script complete with sandhangan swara, with a total of 4800 images. This number covers 120 characters, each consisting of 40 samples. The entire dataset was obtained directly from the researchers' own creations. The image format used is .jpg with a resolution of 99x99 pixels. Figure 2 below is an example from the collected dataset of script images.

Figure 2. Data Collection

2.1.2 Preprocessing

The next step in this research is the preprocessing stage, which aims to improve the quality of the image data to be more optimal during the feature extraction process [15]. The preprocessing stage begins with adding 10-pixel padding on each side to ensure the character's position is consistently in the center of the image. Subsequently, a downsize is performed using the average pooling method until the image size becomes 33x33 pixels. After that, the image is

Building of Informatics, Technology and Science (BITS)
Volume 7, No 2, September 2025 Page: 984−992
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i2.7625

segmented and converted into grayscale format. The final stage of preprocessing is the conversion of the grayscale image to a binary image with the help of thresholding. The result of the preprocessing is shown in Figure 3, which displays a significant change from the raw data to the processed data. In the image before preprocessing, the character is shown in black with a white background. After undergoing the preprocessing stages, the colors are inverted so that the character appears white on a black background.

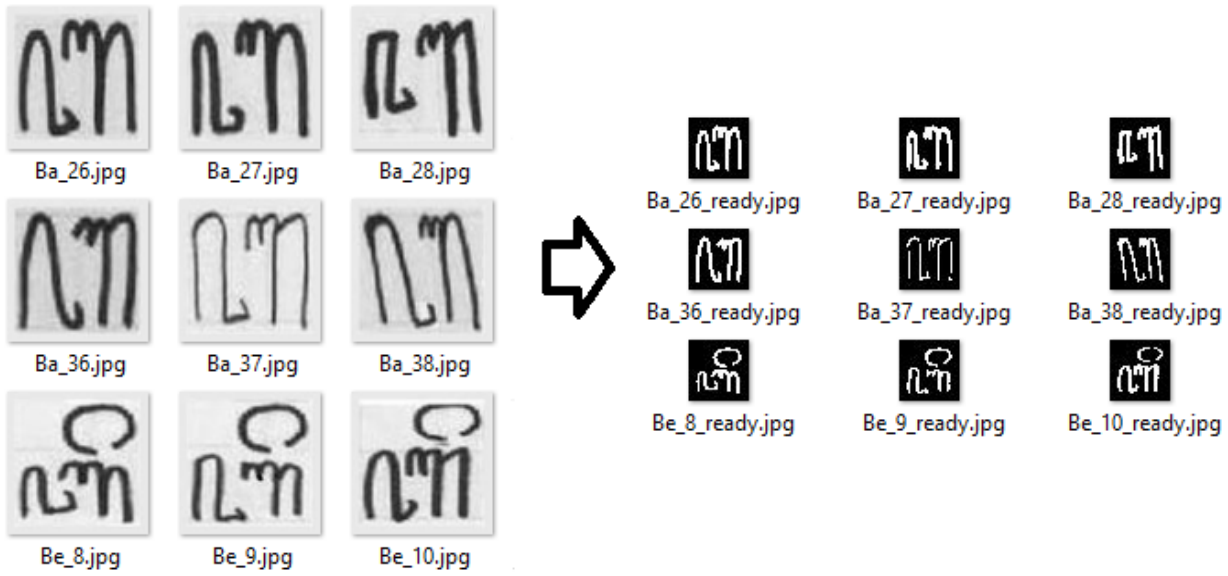

Figure 3. Image Data Before Preprocessing (white background and black character) and After Preprocessing (black background and white character)

2.1.3 Model Training

Model training was conducted using an Artificial Neural Network (ANN) architecture that was built manually without using pre-trained models like CNN or MobileNetV2. This model was trained to recognize 120 classes of characters, each with 40 samples of preprocessed images stored in .npy format. The dataset was then divided in a stratified manner using the Stratified Shuffle Split method with an 80:20 ratio, where 80% of the data was used to train the model and 20% was used to validate the model. Class labels were extracted by taking the argmax of the one-hot label representation, and the splitting process was performed based on these labels. The model has 1 hidden layer with 480 neurons, as well as a total of 1089 input neurons derived from the 33x33 pixel image dimension. The training process was carried out for 500 epochs with a learning rate of 0.001. The activation function used in the hidden layer is sigmoid, chosen for its continuous nature and its ability to produce output in a probabilistic range, which is suitable for classification tasks. As an alternative, the ReLU activation function is also available in the implementation, but the main training used sigmoid.

The model training process was conducted by applying the Stochastic Gradient Descent (SGD) algorithm combined with the backpropagation method. The main objective of this process is to minimize the error between the model's predictions and the target labels by updating the weights and biases at each iteration [16]. The training process is divided into two main stages: forward propagation and backward propagation. In the forward propagation stage, the input data ($\boldsymbol{x}$) is first multiplied by the initial weights from the input to the hidden layer ($\boldsymbol{W_{ih}}$), and then the bias ($\boldsymbol{b_h}$) is added. The result is then processed using the activation function ($\boldsymbol{f}$) (sigmoid in this case) to produce the activation at the hidden layer. This process is formulated as:

$$h_{pre} = W_{ih} \cdot x + b_h \ , \ h = f(h_{pre}) \tag{2}$$

Then, the activation ($\boldsymbol{h}$) is multiplied by the weights from the hidden layer to the output layer ($\boldsymbol{W_{ho}}$), and the bias ($\boldsymbol{b_o}$) is added to produce the final output:

$$o_{pre} = W_{ho} \cdot h + b_o \ , \ o = f(o_{pre}) \tag{3}$$

In the backward propagation stage, the error is calculated from the difference between the predicted output ($\boldsymbol{o}$) and the actual label. This error is referred to as the output delta (δ_o), which is used to calculate the changes to the weights and biases.

$$\delta_o = o - label \tag{3}$$

$$W_{ho} = W_{ho} - \eta \cdot \delta_o \cdot h^T, \quad b_o = b_o - \eta \cdot \delta_o$$

Building of Informatics, Technology and Science (BITS)
Volume 7, No 2, September 2025 Page: 984−992
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i2.7625

The weights W_{ho} are updated by subtracting the product of the learning (η), the output error (δ_o), and the transpose of the hidden layer activation (h^T). The output bias (b_o) is also updated by subtracting the value of the output error. This approach aims to reduce the error by modifying the model's parameters to approach the target.

Then, the error for the hidden layer (δ_h) is calculated by multiplying the output delta by the transposed weights (W_{ho}^T) from the output to the hidden layer and multiplying by the derivative of the activation function ($\acute{f}$) :

$$\delta_h = (W_{ho}^T \cdot \delta_o) \cdot \acute{f}(h) \tag{5}$$

$$W_{ih} = W_{ih} - \eta \cdot \delta_h \cdot x^T, \quad b_h = b_h - \eta \cdot \delta_h$$

The entire training process was conducted using a full-batch approach, where the entire training dataset is used at once in each parameter update iteration. The model does not use external frameworks such as TensorFlow or PyTorch but was implemented entirely manually using NumPy and run on a CPU. During training, the model is run for a number of epochs, where one epoch is one complete cycle of the entire dataset through the network. Each epoch allows the model to adjust its weights to make the prediction error smaller. The learning rate (η) functions as a controller for the rate of weight change: a value that is too large can cause the model to become unstable, while a value that is too small makes convergence very slow.

2.1.4 Model Evaluation

To assess how well the model recognizes Javanese script characters, an evaluation process was conducted using several approaches, including per-class accuracy, a loss function, and a confusion matrix.

a. Per-Class Accuracy
Since this model recognizes 120 classes of characters, it is important to calculate the accuracy of each class individually to determine how well the model recognizes each character. The accuracy for each class is calculated based on the ratio between the number of correct predictions for a specific class (True Positive/TP) and the total amount of data from that class, including those misclassified (False Negative/FN). It is mathematically formulated as:

$$Average_{Accuracy} = \frac{1}{C}\sum_{i=1}^{C} \frac{TP_i}{TP_i + FN_i} \times 100\% \tag{6}$$

where (C) is the total number of classes (in this case, 120 classes), (TP_i) is the number of true positive predictions for the i-th class, and (FN_i) is the number of false negative predictions from the i-th class. The average value of all per-class accuracies provides information about how balanced the model's performance is across all classes.

b. Loss Function – Mean Squared Error (MSE)
During the training process, the model uses Mean Squared Error (MSE) as the loss function to measure the magnitude of the difference between the predicted value and the actual target value. The MSE function is calculated as the average of the squared differences between the actual output (y) and the predicted output (ŷ) [17], formulated as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{7}$$

Where (y_i) is the i-th target output, $\hat{y}_i$ is the i-th predicted output, and n is the number of output neurons. MSE is used to guide the weight update process during training, where a decreasing value indicates that the model is successfully minimizing the overall prediction error.

c. Confusion Matrix
A confusion matrix is an evaluation method used to assess the extent to which a model can correctly make predictions across the classes in the test data [18].

$$\text{ConfMatrix}_{i,j} = \text{Number of data with label } i \text{ predicted as } j \tag{8}$$

where (i) is the actual label (true class) and (j) is the predicted label. The confusion matrix facilitates the tracking of the model's error patterns, especially for classes that have similar handwriting forms, and serves as a basis for further analysis related to model improvement.

2.1.5 Application Development & Model Integration

Application development was carried out to implement the workings of the trained Artificial Neural Network (ANN) model and to visualize its prediction results in the form of a Graphical User Interface (GUI). This application was designed using the PySide6 library. PySide6 is the official library from Qt for Python that provides various modern GUI components such as buttons, labels, image areas, and responsive interactive windows. Compared to Tkinter, PySide6 offers a more modern and flexible appearance for Python-based desktop applications [19]. Model integration was performed by loading the weight and bias parameters from the training results into the application. Users can select a Javanese script image through the interface, then the image is processed through the preprocessing stages before being classified by the ANN model. The classification result is displayed directly in the application.

Building of Informatics, Technology and Science (BITS)
Volume 7, No 2, September 2025 Page: 984−992
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i2.7625

2.1.6 Application Testing

The testing process was conducted to evaluate the application's performance in recognizing Javanese script through images selected manually by the user. A total of 600 additional images, outside of the training and validation test data, were used in this stage, where each script class was represented by 5 samples. Each test image first went through the same preprocessing stages as during training. The preprocessed image was then given to the trained Artificial Neural Network (ANN) model for classification. The model's prediction results were compared with the actual labels to calculate the accuracy, which is the ratio between the number of correct predictions and the total number of test data [20]. This result is used to assess the extent to which the model can generalize to new data in the application environment.

3. RESULT AND DISCUSSION

This section analyzes the model training results and evaluates its performance based on accuracy, loss, and prediction performance for each class of Javanese script characters. The discussion also covers the model's utilization within the application.

3.1 Model Evaluation Results

The model was trained for 500 epochs with 480 hidden neurons and a learning rate of 0.001, using preprocessed and labeled Javanese script image data.

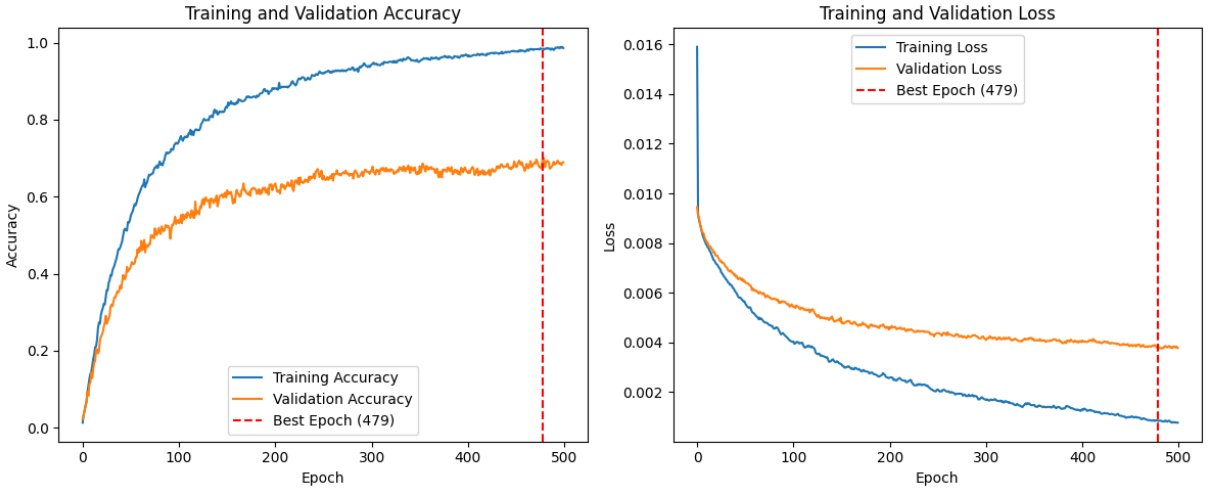

Figure 4. Accuracy and Loss for Epochs 1 to 500

As seen in Figure 4, the validation accuracy fluctuates but shows an increasing trend as the number of epochs increases. The best model was obtained at epoch 479 with a validation accuracy of 70.21%, which indicates the model's ability to generalize to test data it had not seen before. The accuracy and loss curves show a positive trend, where accuracy continuously increases and the loss value decreases with more epochs. This signifies that the model successfully learned important features from the data. However, the considerable gap between the training and validation accuracy indicates that overfitting has occurred, where the model has adapted too closely to the training data and is less optimal when encountering new data. Given that the model was not equipped with augmentation or regularization techniques during training, the stagnant validation performance reflects the need for additional strategies in future model development. Several recommended approaches to improve the model's performance in subsequent training include implementing data augmentation techniques to increase dataset variation, using regularization methods like dropout or L2 regularization to reduce model complexity, and applying an early stopping mechanism to prevent overtraining.

The per-class model evaluation results, as shown in Figure 5, indicate that although the model performs reasonably well on most classes, several classes still have low accuracy. Some Javanese script characters show low accuracy, such as Pe (12%), Dee, Ngee, Ngu, Tee, and Yo (each 25%). Conversely, characters like Tu, Ra, Ca, Gu, and Bo achieved accuracies of up to 100%. This disparity shows that the model has not yet been able to recognize all classes uniformly. The low accuracy for certain characters is likely caused by visual similarity between characters, the complexity of the letter forms, and high variations in handwriting.

Building of Informatics, Technology and Science (BITS)
Volume 7, No 2, September 2025 Page: 984−992
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i2.7625

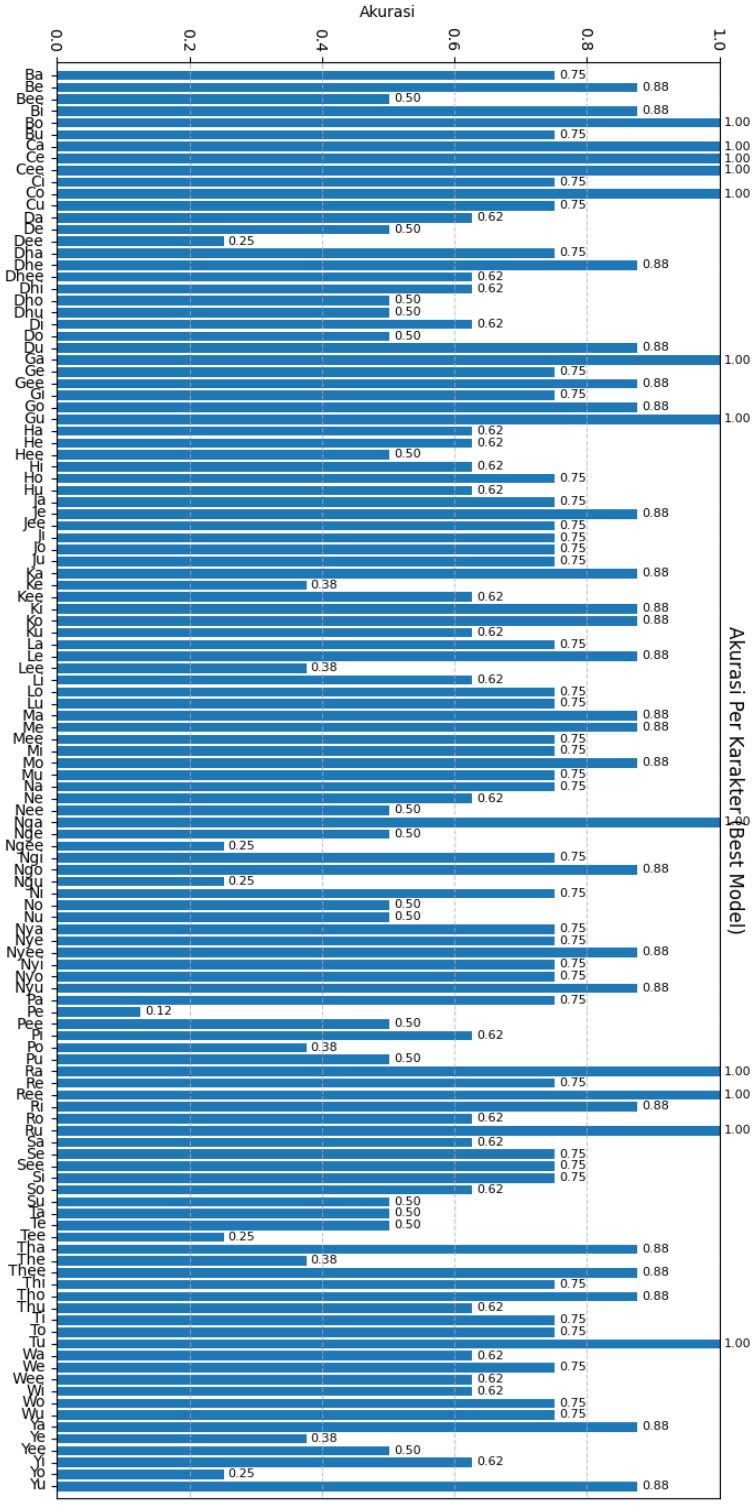

Figure 5. Per-Class Accuracy

The model evaluation process was reinforced by the application of a confusion matrix to identify classification error patterns in more detail. The confusion matrix allows for an analysis of the relationships between classes, especially characters that tend to be confused with one another. Figure 6 below shows the results of the model evaluation based on the confusion matrix, with several significant findings:

a. The character Ngu was only correctly classified in 2 out of 8 samples, while 4 other samples were misclassified as Thu.
b. The character Dee experienced 6 misclassifications, 3 of which were mistaken for Pee.
c. The character Pe was correctly recognized in only 1 sample, while the rest were misclassified as other characters.

Building of Informatics, Technology and Science (BITS)
Volume 7, No 2, September 2025 Page: 984−992
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i2.7625

These findings indicate the existence of visual similarity between characters, which makes it difficult for the model to distinguish between classes consistently. Furthermore, the limited number of samples and the high variation in handwriting further increase the probability of misclassification between similar characters.

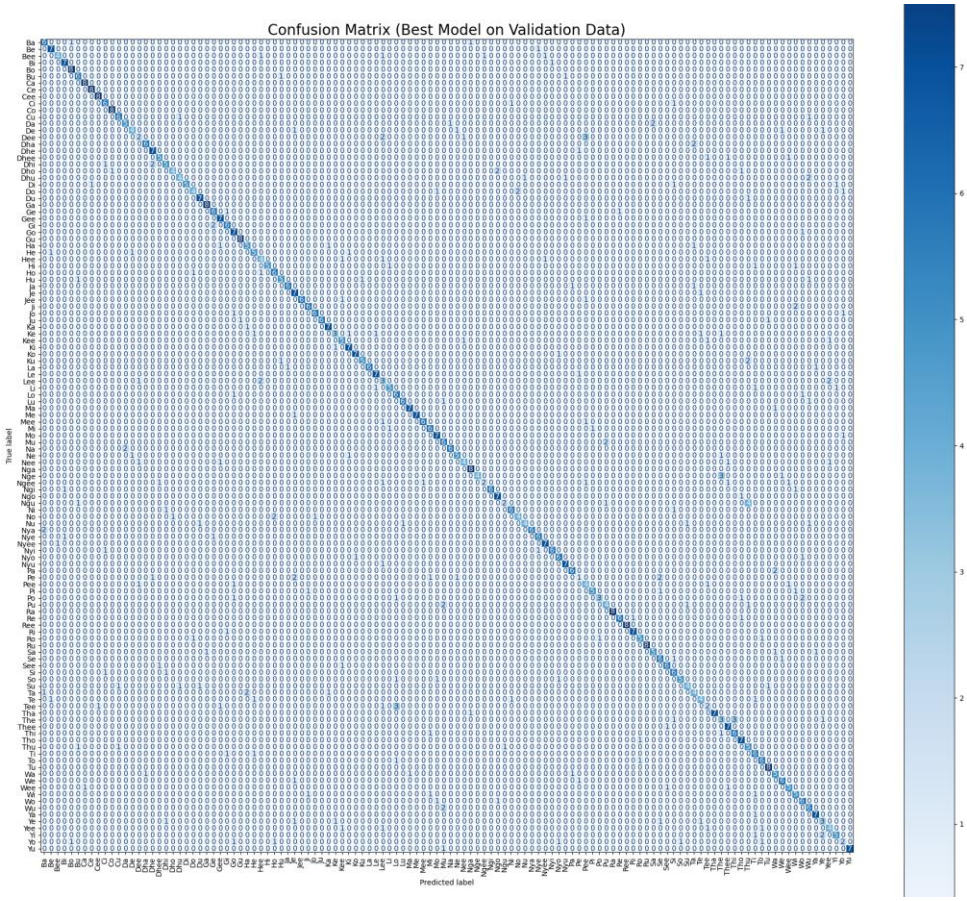

Figure 6. Confusion Matrix for All Classes

Key:
Vertical axis (columns): Shows the true label.
Horizontal axis (rows): Shows the label predicted by the model.

3.2 Application Testing Results

Application testing was conducted in a real-world scenario using test images in JPG format that were manually uploaded by the user. Each image captured is processed through the same preprocessing stages as during training and then classified using the Artificial Neural Network (ANN) model that has been integrated into the application.

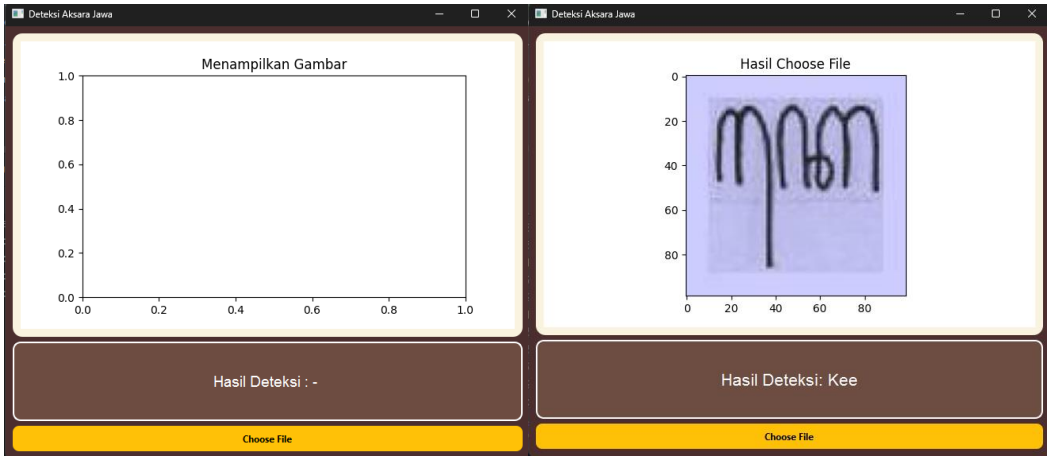

Figure 7. Initial Application View

Figure 8. Application View After File Selection

Building of Informatics, Technology and Science (BITS)
Volume 7, No 2, September 2025 Page: 984−992
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i2.7625

As shown in Figure 7, the initial view of the application provides a simple interface that makes it easy for users to conduct tests. After the user selects an image via the "Choose file" feature, the application displays a preview of the image, and the classification result is shown directly after the recognition process is complete, as seen in Figure 8.

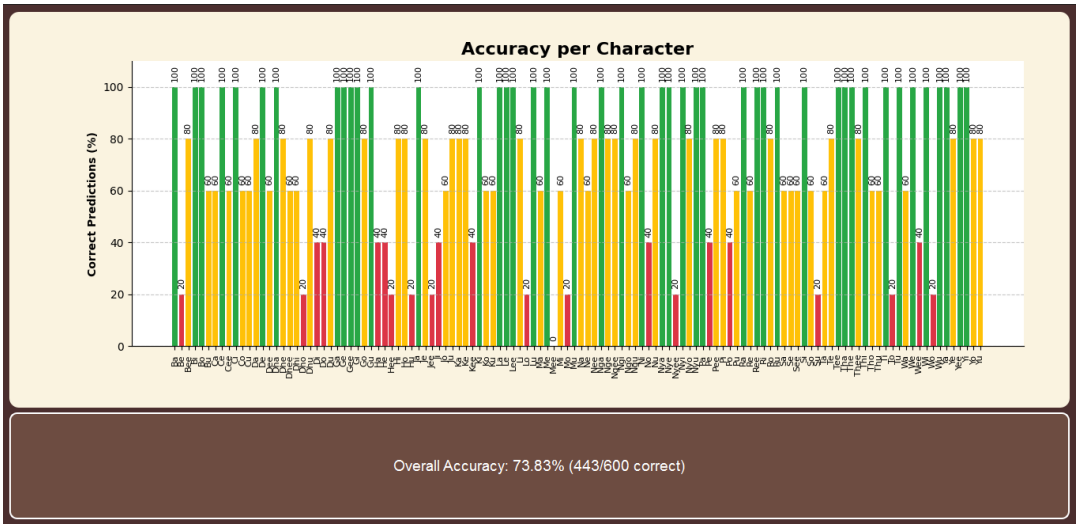

Figure 9. Application Testing Results with Test Data

The application testing results show that the system is able to classify the majority of Javanese scripts with a reasonably good level of accuracy. A total of 443 out of 600 test samples were classified correctly, resulting in an overall accuracy of 73.83%. The test sample consisted of 120 character classes, each represented by 5 new images that had never been used in the training process. The visualization in Figure 9 presents the per-class classification accuracy in the form of a bar graph colored according to the accuracy range:

a. Green (81–100%): indicates very good classification,
b. Yellow (51–80%): indicates fairly good classification,
c. Red (0–50%): indicates low classification.

The majority of classes (more than 60%) are in the green zone, signifying that the model has successfully identified visual patterns quite accurately. Several characters such as Ba, Bi, Bo, Ci, De, Ga, Gi, Gu, Ka, Ke, La, Le, Lu, Me, Mu, Nga, Ngi, Ni, Nya, Nye, Pa, Ra, Ree, Ri, Ru, Si, Tha, The, Thi, Ti, Tu, We, Wi, Wu, Ya, Yee, and Yi achieved 100% accuracy, meaning all images in those classes were classified correctly by the model.

However, there were also several characters that experienced classification difficulties, indicated by the red zone, such as Mee, Hee, Dho, Lo, Mo, Nyee, Su, To, and Wo, all of which scored an accuracy of ≤20%. This indicates the presence of visual similarity between characters, complexity of the handwritten forms, or possible limitations in the representation within the training data. The characters in the yellow zone, such as Pee, Ju, Ko, Ro, Ngo, and Thee, reflect classification results that are fairly good but still have the potential to be improved through data augmentation, improving input quality, or adjusting the model architecture.

4. CONCLUSION

This research has successfully developed an application for real-time Javanese script recognition using an Artificial Neural Network (ANN) algorithm. The ANN model used is capable of classifying Javanese script images reasonably well. The evaluation results show that the model achieved an accuracy of 70.21%, while testing using the application with separate test images obtained an accuracy of 73.83%. However, this application still has several limitations. The testing process is still limited to manual image file selection and does not yet support direct image capture via a camera in real-time. Additionally, data augmentation has not been implemented, which could potentially improve the model's performance. Suggestions for future development are to add real-time camera support for direct testing and to implement data augmentation techniques during the training phase. This would improve the model's generalization ability towards various image variations, such as changes in lighting, viewing angles, rotation, and script size. Data augmentation is crucial so that the model not only recognizes patterns from uniform images but is also able to recognize Javanese script in more varied conditions as found in real-world use.

ACKNOWLEDGMENT

This research was supported by the Center for Urban Studies laboratory and the governance of the Institute for Research and Community Service of Pembangunan Jaya University.

Building of Informatics, Technology and Science (BITS)
Volume 7, No 2, September 2025 Page: 984−992
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i2.7625

REFERENCES

[1] Jonathan and I. Wasito, "Perancangan Aplikasi Pengenalan Aksara Jawa Digital Menggunakan Convulotional Neural Network dan Computer Vision", Decode: Jurnal Pendidikan Teknologi Informasi, vol. 3, no. 2, p. 364–377, 2023, https://doi.org/10.51454/decode.v3i2.209

[2] A. Akhmadi, "Aplikasi pembelajaran aksara jawa berbasis library android gesture recognation dengan menggunakan rule base," Etheses, Malang, 2018.

[3] D. Bram, "Krisis Penggunaan Bahasa Jawa pada Generasi Muda: Mulai Terkikis dari Keluarga," Radar Solo, 20 Maret 2023. [Online]. Available: https://radarsolo.jawapos.com/pendidikan/841700876/krisis-penggunaan-bahasa-jawa-pada-generasi-muda-mulai-terkikis-dari-keluarga. [Accessed 8 Juni 2025].

[4] I. G. S. M. Diyasa and Romadhon, "Klasifikasi Karakter Tulisan Aksara Jawa Menggunakan Algoritma Convolutional Neural Network," in Seminar Keinsinyuran 2023, Surabaya–Malang: Universitas Pembangunan Nasional "Veteran" Jawa Timur dan Universitas Muhammadiyah Malang, 2023, pp. 927–936.

[5] D. Septhian, N. Syafi'al, dkk., "Implementasi Cnn Arsitektur Mobilenetv2 Untuk Klasifikasi Tulisan Aksara Jawa", Seminar Nasional Teknologi & Sains, 298-303, 2024, https://doi.org/10.29407/z1cbjw36

[6] S. Hamzah dan D. P. Pamungkas, "Pengenalan Tulisan Tangan Aksara Jawa Menggunakan Metode Learning Vector Quantization (LVQ) dan Euclidean Distance", inotek, vol. 5, no. 1, pp. 225–230, Aug. 2021, https://doi.org/10.29407/inotek.v5i1.952.

[7] A. Susanto, D. Sinaga, E. H. Rachmawanto, and D. R. I. M. Setiadi, "Unjuk Kerja K-Nearest Neighbors pada Pengengalan Karakter Jawa Berbasis Local Binary Pattern," Seminar Nasional Teknologi Informasi dan Komunikasi (SNATIF), 2019

[8] M. A. Faizin, "Deteksi Aksara Jawa Menggunakan YOLO untuk Transliterasi Berbasis LSTM pada Manuskrip Jawa Kuno," Tugas Akhir, ITS Surabaya, 2023. [Online]. Tersedia: https://repository.its.ac.id/102757/

[9] D. M. H. Ilmawan, B. Warsito, and S. Sugito, "Penerapan Artificial Neural Network Dengan Optimasi Modified Artificial Bee Colony Untuk Meramalkan Harga Bitcoin Terhadap Rupiah," Jurnal Gaussian, vol. 9, no. 2, pp. 135-142, May. 2020, https://doi.org/10.14710/j.gauss.9.2.135-142

[10] A. . Nugraha, Y. . Suparman, and A. . Apriliyanti Pravitasari, "Penerapan Artificial Neural Network Backpropagation untuk Meramalkan Nilai Ekspor Indonesia", *SNS*, vol. 10, p. 37, Dec. 2021, https://doi.org/10.1234/pns.v10i.102

[11] A. F. O. Gaffar, R. Malani and A. B. W. Putra, Artificial Intelligence, Samarinda: MNC Publishing, 2021.

[12] L. Panneerselvam, "Activation Functions Neural Networks: A Quick & Complete Guide", Analytics Vidhya, Juni. 8, 2025. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/04/activation-functions-andtheir-derivatives-a-quick-complete-guide/

[13] A. Ovidius, G. W. Nurcahyo, Sumijan, and R. Salambue, "Akurasi dalam Mengidentifikasi Citra Anggrek Menggunakan Backpropagation Artificial Neural Network", jidt, vol. 3, no. 3, pp. 95-102, Sep. 2021, https://doi.org/10.37034/jidt.v3i3.115

[14] Siregar, M. R., Azhari, A. P., Hartama, D., & Windarto, A. P, "Peramalan Nilai Penjualan Gas Elpiji 3 Kg di Sumatera Utara dengan bantuan Analisis Metode Jaringan Saraf Tiruan", Bulletin of Artificial Intelligence, 1(2), 52-58, Oct. 2020, https://doi.org/10.62866/buai.v1i2.51

[15] U. Rosyidah dan N. Rochmawati, "Analisis Kepribadian Melalui Tulisan Tangan Menggunakan Metode Support Vector Machine," JINACS, vol. I, pp. 91-96, 2019, https://doi.org/10.26740/jinacs.v1n02.p91-96

[16] Z. Rais, Sudarmin, and A. Syahputra, "Backpropagation Neural Network Method For The Classification of Districts/Cities Based On Macro Socio-Economic Indicators In The Province Of South Sulawesi", Quant. Econ. Manag. Stud., vol. 6, no. 2, pp. 273-282, Apr. 2025, https://doi.org/10.35877/454RI.qems3982

[17] J. Anggara, F. R. Ramadhan, dkk., "Pengembangan Sistem Prediksi Harga dan Rekomendasi Mobil Bekas Berbasis Machine Learning", Journal of Technology Informatics (JoTI), Vol.7, No.1, April. 2025, https://doi.org/10.37802/joti.v7i1.987

[18] N. C. I. Natun, M. A. Santhia, dkk., "Identifikasi Pengenalan Wajah Berdasarkan Jenis Kelamin Menggunakan Metode Convolutional Neural Network (CNN)", Journal of Technology and Informatics (JoTI , Vol.6, No.1, Oktober. 2024, https://doi.org/10.37802/joti.v6i1.694

[19] M. Fitzpatrick, Create GUI Applications with Python & Qt6 (PySide6 Edition). Martin Fitzpatrick, 2021. [Online]. Available: https://books.google.co.id/books?id=nfFUEAAAQBAJ

[20] M. Azhari, Z. Situmorang, and R. Rosnelly, "Perbandingan Akurasi, Recall, dan Presisi Klasifikasi pada Algoritma C4.5, Random Forest, SVM dan Naive Bayes", Jurnal Media Informatika Budidarma, vol. 5, no. 2, 2021, https://doi.org/10.30865/mib.v5i2.2937