Convolutional Neural Network Method for Classification of Syllables in Javanese Script

Yuli Fauziah[1], Kevin Aprilianta[2], Heru Cahya Rustamaji[3]*
[1,2,3]Informatics Department, Universitas Pembangunan Nasional "Veteran" Yogyakarta, Yogyakarta, Indonesia
[1]yuli.fauziah@upnyk.ac.id; [2]kevinaprilianta@gmail.com; [3]herucr@upnyk.ac.id*
*Corresponding author

ABSTRACT

Javanese script is one of the languages which are a typical Javanese culture. Javanese script is seen in its use in writing the name of a particular agency or location that has historical and tourism value. The use of Javanese script in public places makes the existence of this script seen by many people, not only by the Javanese people. Some of them have difficulty recognizing the Javanese characters they encounter. One method of pattern recognition and image processing is Convolutional Neural Network (CNN). CNN is a method that uses convolution operations in performing feature extraction on images as a basis for classification. The process consists of initial data processing, classification, and syllable formation. The classification consists of 48 classes covering Javanese script types, namely basic letters (*Carakan*) and voice-modifying scripts (*Sandhangan*). It is tested with multi-class confusion matrix scenarios to determine the accuracy, precision, and recall of the built CNN model. The CNN architecture consists of three convolution layers with max-pooling operations. The training configuration includes a learning rate of 0.0001, and the number of filters for each convolution layer is 32, 64, and 128 filters. The dropout value used is 0.5, and the number of neurons in the fully-connected layer is 1,024 neurons. The average performance value of accuracy reached 87.65%, the average precision value was 88.01%, and the average recall value was 87.70%.

Keywords: Javanese Script, Pattern Recognition, Convolutional Neural Network

Article History

Received :	Revised :	Accepted :

I. INTRODUCTION

The culture of each region in Indonesia has its uniqueness. One of the regional cultures that need to be preserved is the script. In Java, especially in Central Java, there are Javanese scripts which, through Central Java Regional Regulation Number 9 of 2012 concerning Language, Literature, and Javanese Scripts, are sought to be maintained. Efforts to preserve Javanese script can be made by using Javanese script in everyday life and through learning in education [1]. One form of using Javanese script in everyday life is writing the name of the location of a particular agency or place with historical and tourism value. The use of Javanese script in public places makes the existence of this script seen by many people, with some having difficulty recognizing the meaning of the Javanese script they encounter. This difficulty gave rise to the idea of development involving technology to assist the process of detecting and recognizing Javanese characters.

Research related to the detection of the presence of Javanese characters was carried out using the Convolutional Neural Network (CNN) method on objects in the form of images [2]. The research carried out the detection process by forming a bounding box around a series of Javanese scripts encountered. The results of the detection study reached a precision value of 96% and recall of 83% but did not include the recognition of the meaning of each detected Javanese script. Research related to the introduction of Javanese characters was carried out with various approaches. The artificial neural network approach that has been used in the Javanese script recognition research, namely Bidirectional Associative Memory Network, Counter propagation Network, Evolutionary Network, Backpropagation Network, and a combination of Backpropagation with Chi2 [3]. Based on this research, the combination of the Backpropagation method with Chi2 provides the best accuracy reaching 98% for data that has been trained and 73% for data that has not been trained [3]. Another approach to the introduction of Javanese characters is to use machine learning [4]–[6] and deep learning [7]–[10]. Deep learning is used CNN by applying a certain model variation and Deep Neural Network (DNN). Research using the CNN model that refers to the MNIST standard results in a recognition accuracy rate of 94.57%, a precision value of 94.75%, and a recall of 94.57% [7]. Another study by building a CNN model consisting of three convolution layers with filters measuring 3x3 and 2x2 and one fully-connected layer resulted in an average recognition accuracy value of 84.6% by testing using cross-validation [8]. Comparative research between CNN and DNN in classifying Javanese characters produces an accuracy rate of 70.22% on CNN with one convolution layer and 20 filters. In comparison, the DNN produces an accuracy of 64.41% with two hidden layers [9][11].

The pattern recognition approach is also used for Javanese character recognition. The approach is carried out using the $P algorithm [12]. The $P algorithm uses Euclidean distance to find the level of proximity of each Javanese character pattern. The accuracy of the $P algorithm reaches 83% for the recognition of Javanese characters. Various studies related to the introduction of Javanese script only cover the basic type of script (*Carakan*) [3]-[12]. The Javanese script, which has a writing system based on syllable spelling sorting, is limited to the basic type of script (*Carakan*). The syllabic system in Javanese script makes each alphabet consist of one syllable, which can be formed from a combination of basic letters (*Carakan*) and voice-modifying scripts (*Sandhangan*) to change the voice of the script [13] [14].

Research that includes letters other than basic letters (*Carakan*) was carried out using a dictionary-based segmentation method. A study with the dictionary-based segmentation method obtains an accuracy of 91.08% [15]. The research conducted a mapping of all possible syllables formed using a combination of punctuation marks or voice-modifying characters (*Sandhangan*) with basic letters (*Carakan*) to have a high level of complexity. Another study uses a machine learning approach in the form of a combination of dictionary-based word segmentation methods, Support Vector Machine (SVM), and Conditional Random Field (CRF) [16]. The results achieved a precision level of 86.7%, a recall rate of 94.4%, and an F-1 value of 90.4% in a combination of dictionary-based word segmentation methods and CRF [14].

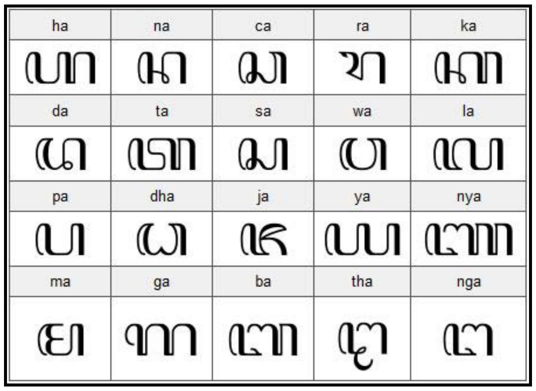
This study will classify the Javanese script using the CNN method approach. The object of the research includes basic letters (*Carakan*) and voice-modifying characters (*Sandhangan*). Scripts that have been classified will be assembled into syllables using a rule base according to the applicable Javanese script guidelines.

II. METHOD

A. Data Collection

The data used as a dataset is a collection of Javanese scripts with basic script types (*Carakan*) and voice-modifying characters (Sandhangan). The voice-changing characters (*Sandhangan*) used include Sandhangan Swara, namely *wulu (i)*, *suku (u)*, *taling (é)*, *pepet (è)* dan *taling tarung (o)* as well as *Sandhangan Panyigeging Wanda* which includes *wignyan (h)*, *layar (r)*, and *cecak (ng)*.

The dataset uses primary data in a questionnaire with a direct survey to 75 respondents and is made independently through the font or Unicode Javanese script. The character classes used are 48 classes. The number of character data obtained is 200 images for each character class or a total of 9,600-character images. An example of the character class used in this study is presented in Fig. 1 [17]. The image data collected is then divided into two parts: data for training and testing. The distribution of the data uses data splitting techniques. The distribution of training and testing data can use 80% for training data and 20% for test data.



ha	na	ca	ra	ka
ꦲ	ꦤ	ꦕ	ꦫ	ꦏ
da	ta	sa	wa	la
ꦢ	ꦠ	ꦱ	ꦮ	ꦭ
pa	dha	ja	ya	nya
ꦥ	ꦝ	ꦗ	ꦪ	ꦚ
ma	ga	ba	tha	nga
ꦩ	ꦒ	ꦧ	ꦛ	ꦔ

Fig. 1. *Carakan* Javanese script

B. Javanese script Detection Process

Path search is carried out with the use of the BFS algorithm as a solution. Numerous steps must be completed to find this path. Fig. 3 depicts a flowchart illustrating all of the phases involved in the pathfinding process. It will be easier for researchers to incorporate this diagram into their applications if they start with this one. The stages of the analysis of the Javanese script detection process are presented in Fig. 2.

Fig. 3 is the process of checking the background color of the image to anticipate if the input image to be tested has a background color close to white. After that, the background color will be changed to black. Therefore, all input images will have a black background color. It converts the input image (input) in the form of RGB into a grayscale image. This conversion process is carried out to facilitate the following image processing process. Grayscale images with the only one-color channel will be easier and simpler to process than RGB images with three color channels, namely Red, Green, and Blue. The stage of the character detection process is to perform morphological operations in the form of dilation. This dilation operation is carried out to thicken the area in an image. This is done to anticipate if there are objects that are cut off due to image capture disturbed by noise, damage to physical objects, or poor resolution such as damaged paper so that the object obtained is cut off.

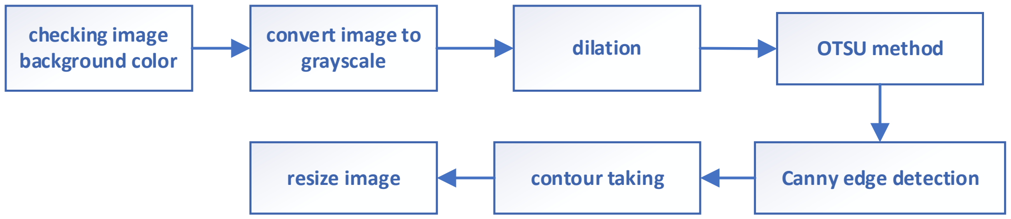


Fig. 2. Javanese Script Detection Process

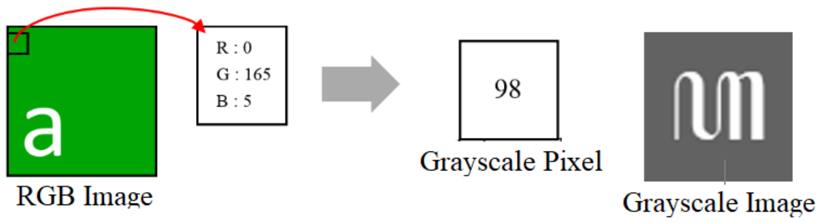


Fig. 3. Convert Grayscale Image

The next step is to find the threshold value using the Otsu method. The Otsu method aims to obtain a dynamic threshold value or adjust to the processed input image. Thus, the threshold value does not become an obstacle for the following process. Constraints in question, such as the possibility of losing vital information in the image if the threshold value is set too high, are discussed further below. On the other hand, if the threshold value is too low, it can cause edge detection errors in the image.

The input image consisting of more than one character object makes the need for edge detection to separate each object. Canny edge detection will obtain the appearance of the boundary line of an object in the image. This boundary line will determine whether an object is detected. The Canny edge detection process in this study uses a dynamic threshold value. The Otsu method is used to calculate the threshold value, which is determined by the results of the operation that has been performed earlier. The use of this threshold value lies in the hysterical threshold processing section. In the hysterical threshold, two threshold values are used, namely the upper threshold and lower threshold. The threshold value obtained from the operation of the Otsu method becomes the upper threshold, and the lower threshold value is obtained from half the upper threshold value. The results of Canny edge detection in the form of boundary lines become the basis for detecting every character object contained in the image.

Each character object that has been identified in the image, then its contour is detected. Detection of the contours of each object is done to get a new image by cutting the image according to the existing contour. Image cropping is done by forming a square at the four outermost points of the contour. This is done as the beginning of the process of separating objects between the basic letters (*Carakan*) and the sound-changing characters (*Sandhangan*).

The image resizing process in this study aims to make the size of the entire contour detection image 64x64 pixels. This image resizing is only to simplify the classification process in CNN and does not change or eliminate information in the form of essential pixels in the image.

C. Analysis Of Syllable Formation in Javanese Script

The flowchart in Fig. 4 is the flow of syllable formation from each character object through the classification stage. In general, syllables in Javanese script can be formed independently by only the basic letters. However, in its use, some syllables undergo certain voice changes. This voice changes, which in the writing rules requires a voice change character. In this study, if the object of the letter is a basic letter, then the result of the formation of syllables is the character itself without changing the voice. However, if voice-modifying characters are discovered throughout the process, the outcomes of the generation of syllables will cause the script's voice to be altered.

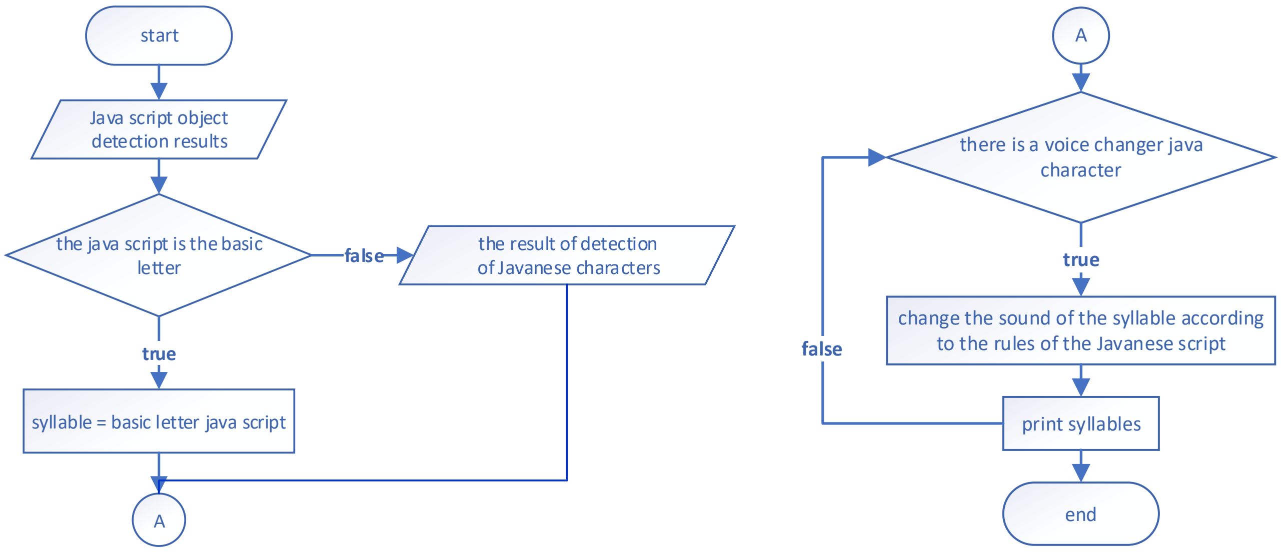

start
Java script object detection results
the java script is the basic letter
false
the result of detection of Javanese characters
true
syllable = basic letter java script
A
A
there is a voice changer java character
true
change the sound of the syllable according to the rules of the Javanese script
false
print syllables
end

Fig. 4. Flowchart of the formation of Javanese script syllables

Table I contains examples of syllable production in various contexts. Table II contains additional criteria for the generation of syllables in Javanese script utilizing voice-modifying characters and basic letters, in addition to the ones listed above.

TABLE I
EXAMPLES OF SYLLABLE FORMATION

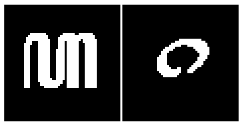
Javanese Script Image	Image Classification Results	Syllable Forming Results
		Base letter: HA Voice changer characters: - Syllable formed: HA
		Base letter: HA Voice changer character: PEPET Syllables formed: HA + PEPET (e) = HE

TABLE II
EXAMPLES OF SYLLABLE FORMATION RULES FOR VOICE-CHANGING CHARACTERS

Javanese Script Voice Changer	Javanese Character Name Voice Changer	New Voice	Description of Use of Syllable Forming
◌ꦶ	*Wulu*	*i*	Change the vowel 'a' to 'i'
◌ꦸ	*Suku*	*u*	Change the vowel 'a' in the base letter to 'u'
ꦺ◌	*Taling*	*é*	Change the vowel 'a' in the base letter to 'e'
◌ꦼ	*Pepet*	*è*	Change the vowel 'a' in the base letter to 'e'
ꦺꦴ ◌	*Taling Tarung*	*o*	Change the vowel 'a' in the base letter to 'o'
◌ꦃ	*Wignyan*	*h*	Adding the letter 'h' at the end of the base letter
◌ꦂ	*Layar*	*r*	Adding the letter 'r' at the end of the base letter
◌ꦁ	*Cecak*	*ng*	Adding the letter 'ng' at the end of the base letter

D. CNN Modelling

The CNN model in Fig. 5 consists of the stages of creating a dataset: the initial data processing process and the CNN architecture analysis that will be used. Initial data processing aims to prepare the dataset that will be used for the modeling process. This processing is carried out on the data obtained through questionnaires filled out by respondents.

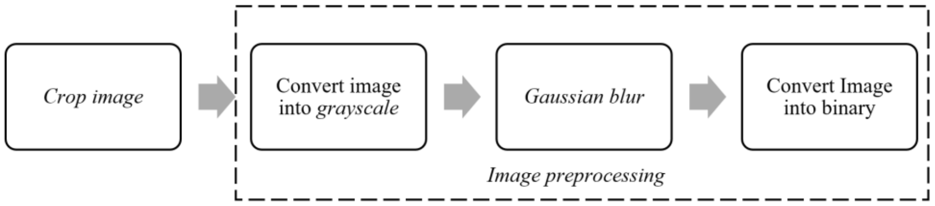


Fig. 5. Data processing

Data derived from the respondent's questionnaire will be scanned manually for each sheet, with the results in Fig. 6. If the scanning process has been carried out, then proceed with cropping the image to a size of 64x64 pixels. The size of this image cropping was chosen to simplify and speed up the modeling process. The part that is cut to be taken as data is the part of the box totaling two for each type of character-filled in by the respondent's handwriting.

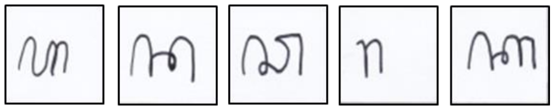


Fig. 6. Image Cropping

Converting to a binary image is done by changing the white part to black and vice versa (inverse). An example of the image preprocessing results from the previously cropped image is presented in Fig. 7.

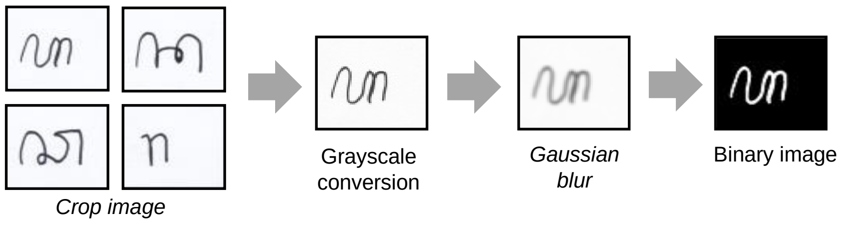


Fig. 7. Illustration Of Data Processing

Initial data processing is image preprocessing. This stage aims to reduce the noise that appears in the previously cropped image. Noise reduction is carried out in three stages: converting the image to grayscale, modifying the image with a Gaussian filter, and converting the existing image into a binary image. The conversion process is carried out by equation (1). Where the $G_{rata-rata}$ variable is the converted pixel value, the R-value represents the red component (Red), the G value is the green component (Green), and the B value represents the blue component (Blue).

$$G_{rata-rata} = 0.299 * \mathrm{R} + 0.587 * \mathrm{G} + 0.114 * \mathrm{B} \tag{1}$$

The next step is to perform a Gaussian filter to remove normally distributed noise (Equation 2). The noise that occurs in the initial data processing can be caused by the reflection of light from scanning images from the questionnaire. The Gaussian filter used is 5x5 with $g(x, y)$ a grayscale image binary image $f(x, y)$ and T the variable threshold value. The pixel value will be one of the input pixel values that is more than the threshold value and will be 0 if the input pixel value is below the threshold value

$$g(x, y) = \begin{Bmatrix} 1\ if\ f(x, y)\ \geq T \\ 0\ if\ f(x, y)\ \leq\ T \end{Bmatrix} \tag{2}$$

The CNN architecture that is built consists of three convolution layers accompanied by max-pooling operations on each layer. Each layer in the convolution layer consists of a filter of a certain size. Filters in CNN are likened to weight values whose values are continually updated. The filter in this study consisted of the first two filters measuring 5x5 and one filter measuring 3x3. Each filter used has a different filter value. The value size given follows the filter size that has been determined, namely 5x5 for the first two filters and 3x3 for the last filter. Another thing related to filters is the presence of strides that determine the convolution shift process in each layer. The stride used in the convolution process is one, which means that the shift is done once to the right until the last pixel and continues downwards until the pixel ends. The equation for the convolution operation is presented in equation (3).

$$A_j = f\left(\sum_{i=1}^{N} I_i * K_{i,j} + B_j\right) \tag{3}$$

Where the A_j variable is each element of the result of the convolution operation. The convolution operation is performed by adding up the product of the input matrix I_i, the kernel $K_{i,j}$, and the kernel and adding bias B_j to each element in the convolution result. These results are then used as input for the f function is namely the ReLU activation function. The equation for the ReLU activation function is presented in equation (4) [8].

$$f(x) = \begin{cases} x\ (x \geq 0) \\ 0\ (x < 0) \end{cases} \tag{4}$$

The results of the convolution operation are then flattened so that they can be used as input for the classification layer. The classification layer in question is a fully-connected layer consisting of several neurons. Each neuron performs a multiplication operation between the input matrix and the internal weight, as presented in equation (5). Where the b_j variable is the bias value, the b_j variable is the number of neurons, and the f variable is the activation function defined in equation (3).

$$u_j = f\left(\sum_{i=1}^{N} w_{j,i}\, x_i + b_j\right) \tag{5}$$

The fully-connected layer used in this study consists of 1,024 neurons equipped with a dropout layer. The dropout layer is a layer that serves to anticipate overfitting. The dropout layer is also used to improve the performance of the artificial neural network for various datasets [18]. The next layer is the SoftMax layer. The SoftMax layer generates the class prediction class distribution. The equation of the SoftMax layer is presented in equation (6) [8].

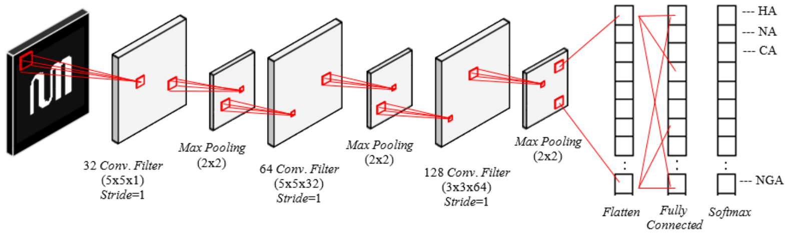
$$p(x) = \frac{e^x}{\sum_{k=1}^{K} e^x} \tag{6}$$

Each architecture in the CNN model that is built has a difference in the number of convolution layers used [7]. The number of convolution layers used in this research is three layers with two filters measuring 5x5 and one filter measuring 3x3. The convolution operation in each layer is accompanied by a stride operation once. In addition, the ReLU activation function is performed for each layer to anticipate negative values in the results of the convolution operation. An illustration of the CNN model architecture that was built in this study is presented in Fig. 8.

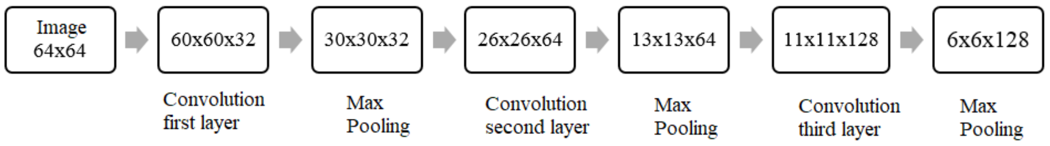
The operation result in Fig.8 from the convolution layer on the CNN architecture is a multi-dimensional array. This multi-dimensional array form needs to be converted into vector form. Changes are made to facilitate the next process, namely the operation process on the fully connected layer that requires input in the form of vectors. The process of changing this shape is called flattening or reshaping feature maps. The input from the flatten process is determined based on the size of the multi-dimensional array as the result of the operation on the last convolution layer after the max-pooling operation, which in this study is 8x8x128.

After the flattening stage is carried out, the next process is in the fully connected layer, which acts as the classification layer. The fully connected layers used in this research are two fully connected layers. The first fully connected layer has a total of 1.024 neurons. Before the process on the second fully connected layer is carried out, a dropout layer functions to fight to overfit. The input from the dropout layer is the result of the first fully connected layer that has gone through the ReLU activation function process. The result of this dropout layer then becomes the input for the second fully connected layer. The second fully connected layer is useful for mapping each classification result into predefined classes. The mapping process is carried out based on the input results from the dropout layer. The mapping process is carried out by continuously updating the existing weight and bias values. The definition of the SoftMax layer has a function to display the probability distribution of the predicted class. The prediction results that have been mapped are calculated with the probability level and sorted from the highest probability level. This SoftMax layer functions when the training results are stored as a model read on the mobile application as a CNN model testing medium.

E-ISSN : 2686-6269



The result of the convolution operation before adding padding (in pixels):



The result of the convolution operation after adding padding (in pixels):

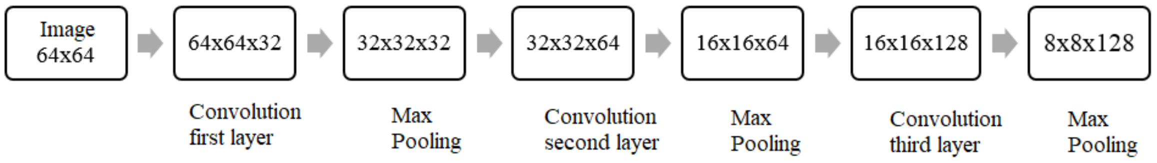


Fig. 8. CNN Model Architecture

III. RESULT AND DISCUSSION

The classification results from each image entered into the CNN model form a class probability distribution consisting of five label classes. Based on this, the classification results are taken with the highest probability value. Taking the result with the highest probability value is assumed that the classification result is the most appropriate result from the input image. The classification results taken are still in the form of labels for certain classes. The label then needs to be defined to find out the meaning of each detected character for each object in the input image. This is a process that is quite decisive in the process of forming syllable letters.

The training process carried out is also equipped with a loss function. The loss function is a function that is used to measure the performance of the neural network in predicting the target. The loss function used is cross-entropy, considering that the scope of this research is on classification problems. Cross entropy calculates the loss of the actual probability distribution that has been carried out through the SoftMax layer. The loss value in the training process can be minimized by using optimization. Optimization is carried out based on computational scalability in CNN. The optimization used is the Adam optimization algorithm with a learning rate of 0.0001. CNN model architecture is built using TensorFlow [19]. TensorFlow is an open-source Python library for deep learning. The computations used in this study consist of an Intel Core i7-6700HQ CPU and an Nvidia GTX 950M GPU. The processing results of each layer on the CNN model architecture are presented in Table III, including the resulting image's dimensions and the details of the filter size used.

TABLE III
CNN MODEL ARCHITECTURE

Layer Type	Size	Output
Input image	(1, 64, 64)	-
Convolution Layer + ReLU	32 (5 x 5) filter	(32, 64, 64)
Max-Pooling	(2 x 2) filter	(32, 32, 32)
Convolution Layer + ReLU	64 (5 x 5) filter	(64, 32, 32)
Max-Pooling	(2 x 2) filter	(64, 16, 16)
Convolution Layer + ReLU	128 (3 x 3) filter	(128, 16, 16)
Max-Pooling	(2 x 2) filter	(128, 8, 8)
Fully-Connected Layer + ReLU + Dropout Layer	1.024 neuron	48
SoftMax Layer	48 class	48

The process of defining characters based on the classification results consists of defining basic letters (*Carakan*) and voice-changing characters (*Sandhangan*). The confusion matrix test is a test carried out to see the performance of a classification model

E-ISSN : 2686-6269

[20]. Performance measurement is done by looking at the parameter values of accuracy, precision, and recall. These parameters are calculated based on the values contained in the confusion matrix, including the True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) values. The confusion matrix used for testing in this study is a multi-class confusion matrix. This is due to the large number of character classification classes used. The total classification class is 48 classes with the results of the confusion matrix in Figure 9.

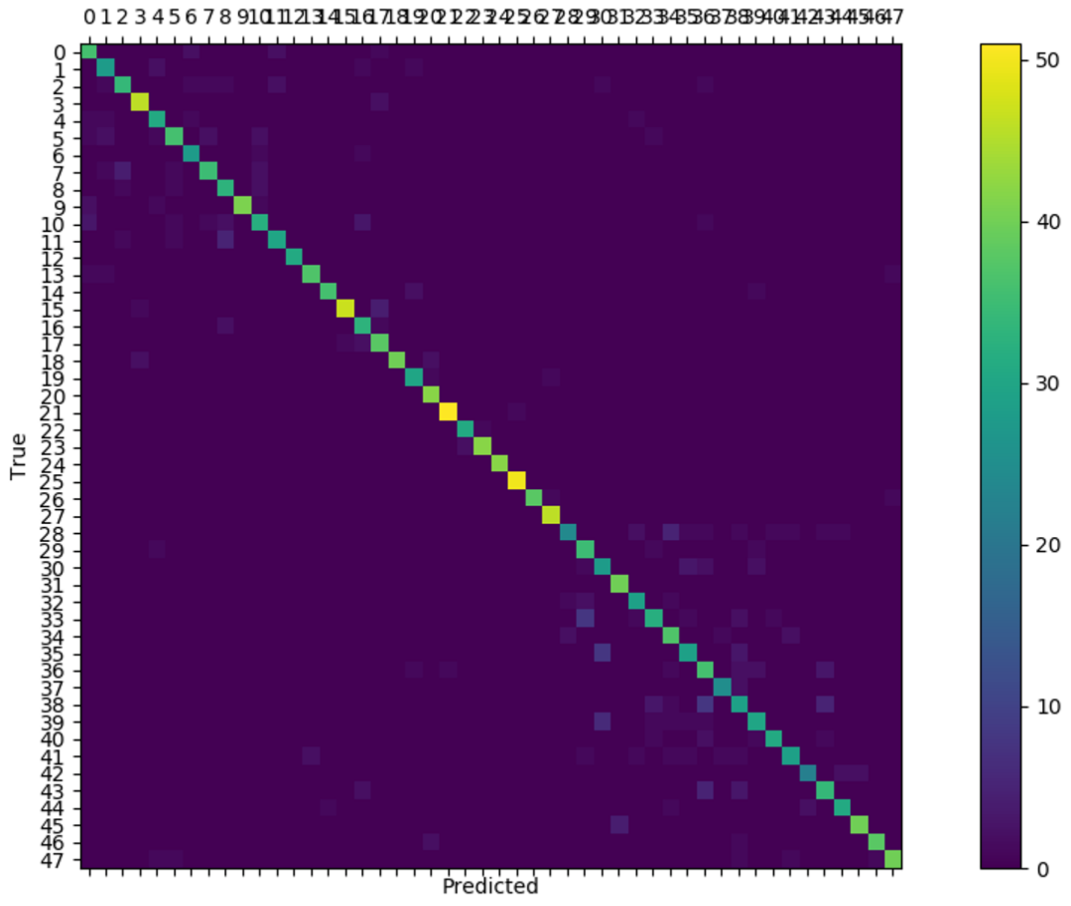


Fig. 9. Confusion Matrix

The confusion matrix test is carried out using test data consisting of 1,920 data. The total test data results from a proportion that is 20% of the total data collected. The distribution of test data for each class is carried out randomly so that it is possible to have differences in the amount of test data for each class of letters. The difference in the amount of test data for each class simultaneously tests the level of accuracy, precision, and recall of the CNN classification model that has been formed for unbalanced data (balance data). The results of the three-parameter test for each data fold are presented in Table IV.

TABLE IV
MEASURE RESULTS OF CNN MODEL ARCHITECTURE

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Accuracy	85.93%	86.35%	86.25%	87.55%	87.65%
Precision	86.50%	86.29%	86.27%	87.78%	88.01%
Recall	86.45%	86.38%	86.45%	87.49%	87.70%

Cross-validation testing is carried out using the confusion matrix facility on Tensorflow. Based on Table 4, the optimal testing results on fold 5 are obtained with an accuracy value of 87.65%, a precision of 88.01%, and a recall of 87.70%. From the results that are relatively even and quite high between the accuracy that occurs with precision and recall, it can be interpreted that it is not overfitting, and is quite stable.

IV. CONCLUSION

The Convolutional Neural Network (CNN) method approach can recognize patterns from Javanese characters and classify the types of each character with test results at an accuracy rate of 87.65%, an average precision value of 88.01%, and an average recall value of 87. ,70%. The test was carried out with 1,920 randomly distributed test data. Accurate classification results play an essential role in the syllable formation process. The formation of syllables is carried out based on the rules of writing Javanese script by considering the presence of a sound-modifying script (*Sandhangan*) that accompanies the basic letter script (*Carakan*).

REFERENCES

[1] E. Nurhayati, Mulyana, H. Mulyani, and Suwardi, “Strategi Pemertahanan Bahasa Jawa di Provinsi Daerah Istimewa Yogyakarta,” *LITERA J. Penelit. Bahasa, Sastra, dan Pengajarannya*, vol. 12, no. 1, pp. 159–166, 2013.
[2] M. L. Afakh, A. Risnumawan, M. E. Anggraeni, M. N. Tamara, and E. S. Ningrum, “Aksara jawa text detection in scene images using convolutional neural network,” in *Proceedings - International Electronics Symposium on Knowledge Creation and Intelligent Computing, IES-KCIC 2017*, 2017, vol. 2017-Janua, pp. 77–82.
[3] G. S. Budhi and R. Adipranata, “Handwritten javanese character recognition using several artificial neural network methods,” *J. ICT Res. Appl.*, vol. 8, no. 3, pp. 195–212, Aug. 2015.
[4] C. A. Sari, M. W. Kuncoro, D. R. I. M. Setiadi, and E. H. Rachmawanto, “Roundness and eccentricity feature extraction for Javanese handwritten character recognition based on K-nearest neighbor,” *2018 Int. Semin. Res. Inf. Technol. Intell. Syst. ISRITI 2018*, pp. 5–10, Nov. 2018.
[5] Y. Sugianela and N. Suciati, “Javanese Document Image Recognition Using Multiclass Support Vector Machine,” *CommIT (Communication Inf. Technol. J.*, vol. 13, no. 1, pp. 25–30, May 2019.
[6] I. F. Katili, F. D. Esabella, and A. Luthfiarta, “Pattern Recognition Of Javanese Letter Using Template Matching Correlation Method,” *J. Appl. Intell. Syst.*, vol. 3, no. 2, pp. 49–56, Dec. 2018.
[7] M. A. Wibowo, M. Soleh, W. Pradani, A. N. Hidayanto, and A. M. Arymurthy, “Handwritten Javanese Character Recognition using Descriminative Deep Learning Technique,” *Inf. Technol. Inf. Syst. Electr. Eng.*, vol. 2, pp. 325–330, 2017.
[8] C. K. Dewa, A. L. Fadhilah, and A. Afiahayati, “Convolutional Neural Networks for Handwritten Javanese Character Recognition,” *IJCCS (Indonesian J. Comput. Cybern. Syst.*, vol. 12, no. 1, pp. 83–94, Jan. 2018.
[9] Rismiyati, Khadijah, and A. Nurhadiyatna, “Deep learning for handwritten Javanese character recognition,” in *Proceedings - 2017 1st International Conference on Informatics and Computational Sciences, ICICoS 2017*, 2017, vol. 2018-January, pp. 59–63.
[10] Rismiyati, Khadijah, and D. E. Riyanto, “HOG and Zone Base Features for Handwritten Javanese Character Classification,” in *2018 2nd International Conference on Informatics and Computational Sciences (ICICoS)*, 2018, pp. 1–5.
[11] Rismiyati, Khadijah, and A. Nurhadiyatna, “Deep Learning for Handwritten Javanese Character Recognition,” *Informatics Comput. Sci.*, vol. 1, pp. 59–64, 2017.
[12] L. L. Zhangrila, “Accuracy Level of $p Algorithm for Javanese Script Detection on Android-Based Application,” in *Procedia Computer Science*, 2018, vol. 135, pp. 416–424.
[13] Darusuprapta, “Pedoman Penulisan Aksara Jawa.” Yayasan Pustaka Nusatama, Yogyakarta, p. 69, 2002.
[14] D. Tanaya and M. Adriani, “Word Segmentation for Javanese Character Using Dictionary, SVM, and CRF,” in *Proceedings of the 2018 International Conference on Asian Language Processing, IALP 2018*, 2019, pp. 240–243.
[15] D. Tanaya and M. Adriani, “Dictionary-based Word Segmentation for Javanese,” *Procedia Comput. Sci.*, vol. 81, pp. 208–213, 2016.
[16] D. Tanaya and M. Adriani, “Word Segmentation for Javanese Character using Dictionary , SVM , and CRF,” *Int. Conf. Asian Lang. Process.*, pp. 240–243, 2018.
[17] G. S. Budhi and R. Adipranata, “Handwritten Javanese Character Recognition Using Several Artificial Neural Network Methods,” *J. ICT Res. Appl.*, vol. 8, no. 3, pp. 195–212, Mar. 2015.
[18] N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” 2014.
[19] T. Developers, “TensorFlow,” Nov. 2021.
[20] K. M. Ting, “Confusion Matrix,” *Encycl. Mach. Learn. Data Min.*, pp. 260–260, 2017.