Handwritten Javanese Character Recognition Using Several Artificial Neural Network Method

by Gregorius Satia Budhi

Submission date: 27-Nov-2019 02:37PM (UTC+0700)
Submission ID: 1222721067
File name: r_Recognition_Using_Several_Artificial_Neural_Network_Method.pdf (571.98K)
Word count: 5386
Character count: 28192

Handwritten Javanese Character Recognition Using Several Artificial Neural Network Methods

Gregorius Satia Budhi, Rudy Adipranata

Informatics Department, Petra Christian University
Jalan Siwalankerto 121-131, Surabaya 60236, Indonesia
Email: greg@petra.ac.id

Abstract. Javanese characters are traditional characters that are used to write the Javanese language. The Javanese language is a language used by many people on the island of Java, Indonesia. The use of Javanese characters is diminishing more and more because of the difficulty of studying the Javanese characters themselves. The Javanese character set consists of basic characters, numbers, complementary characters, and so on. In this research we have developed a system to recognize Javanese characters. Input for the system is a digital image containing several handwritten Javanese characters. Preprocessing and segmentation are performed on the input image to get each character. For each character, feature extraction is done using the ICZ-ZCZ method. The output from feature extraction will become input for an artificial neural network. We used several artificial neural networks, namely a bidirectional associative memory network, a counterpropagation network, an evolutionary network, a backpropagation network, and a backpropagation network combined with chi2. From the experimental results it can be seen that the combination of chi2 and backpropagation achieved better recognition accuracy than the other methods.

Keywords: *backpropagation; bidirectional associative memory; chi2; counterpropagation; evolutionary neural network; Javanese character recognition.*

1 Introduction

Many people on the island of Java use the Javanese language in their conversation. The Javanese language has its own letterforms that differ from Roman characters. Javanese character recognition has its own difficulties because of the shapes of the basic characters, vowels, complementary characters, and so on. Because the characters are difficult to recognize, not many people can read or write Javanese script any more. For many people, Javanese characters will eventually be regarded as decoration only and not mean anything. This will gradually erode the existence of Javanese characters and will ultimately also affect Javanese culture in general.

In this research, we have developed a system that can automatically recognize Javanese characters in the form of a digital image and convert them into digital

Received October 28th, 2014, Revised February 2nd, 2015, Accepted for publication February 13th, 2015.
 ISSN: 2337-5787, DOI: 10.5614/itbj.ict.res.appl.2015.8.3.2

text using a *Hanacaraka* font. The first process is digital image preprocessing, followed by segmentation and feature extraction. The features will be used as input for the recognition system. In this research we used and compared five methods of artificial neural networks for recognition, namely a bidirectional associative memory network, a counterpropagation network, an evolutionary neural network, a backpropagation neural network, and a combination of chi2 and a backpropagation neural network.

In addition to serving as the basis of further research, we hope that with this system the Javanese characters can be preserved and studied more easily. The system, which makes it easy to save articles in Javanese characters to electronic documents, can also help teachers teach the Javanese language in schools. And finally for those who do not know about Javanese script, this system can be used to identify and interpret writings in Javanese characters that they encounter in touristic sites.

2 Related Works

Some researchers have conducted research on Javanese character recognition. Nurmila [1] used a backpropagation neural network. The accuracy rate was about 61%. Another researcher, Priyatma, used fuzzy logic [2] and the recognition results were satisfactory. Also in relation to Javanese character studies, Rudy, *et al*. [3] have developed an application that translates the results of typing on a Roman letter keyboard into Javanese characters using a *Hanacaraka* font. This application can later be combined with the result of the present research to create optical character recognition and an editor for Javanese characters. This research is an extension of previous research [4],[5], which evaluated the use of a backpropagation neural network for character recognition and some improvement in digital image preprocessing.

Several studies on the implementation of artificial neural networks have also been done, including automatic classification of sunspot groups for space weather analysis [6] using backpropagation and other neural network methods. Other studies have used backpropagation methods to recognize types of automobiles [7], to identify abnormalities of the pancreas through the image of the iris with a recognition rate of more than 90% [8], and to recognize characters in a digital image [9].

3 Javanese characters

Compared to Roman characters, Javanese characters have a different structure and shape. The basis of the Javanese characters is called *carakan*, which consists of 20 syllables called *dentawyanjana*, see Figure 1 [10].

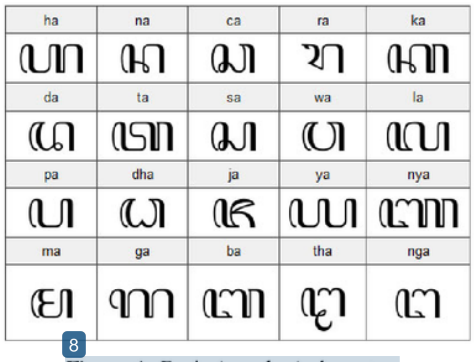

Figure 1 Basic (*carakan*) characters.

Numbers in Javanese characters are shown in Figure 2 [11].

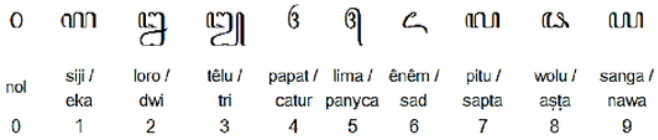

Figure 2 Javanese symbols for numbers.

Sandhangan characters are special characters that are used as complementary characters, vowels or consonants that are commonly used in everyday language. *Sandhangan* can be seen in Table 1 [12].

Table 1 *Sandhangan* characters.

Sandhangan name	Java character	Description
Pepet		*Vowel ê*
Taling		*Vowel é*
Wulu		*Vowel i*
Taling tarung		*Vowel o*
Suku		*Vowel u*

4 Image Segmentation

One of the important processes used to transform an input image into an output image is segmentation. Segmentation is based on the attributes of the image. The image is divided into several regions based on their intensity, so objects and background can be distinguished. Once each object has been isolated or

made clearly visible, segmentation should be discontinued [13]. In this research, we used thresholding and skeletonizing for the segmentation process.

Thresholding is one way of separating objects (foreground) in an image from the background by selecting a threshold value T. In thresholding, the value of T is used to separate all points (x, y) in an image in two categories. All points (x,y), where $f(x, y) > T$, can be called either an object point or a background point [13].

To get rid of redundant pixels and produce an image that is more modest in size, the skeletonizing process is used. The goal of skeletonizing is to make a simpler image, so that the shape and suitability of the image can be analyzed further. The most important problem addressed in the skeletonizing process is how to determine the redundant pixels. If we do not adequately determine the redundant pixels, the skeletonizing process is likely to turn into an erosion process, which can cause image regions to be deleted. The skeleton should have some basic properties, such as [14]:

1. The pixels that form the skeleton should be located near the middle area of the region's cross section.
2. It must consist of several thin regions, with each region having a width of only 1 pixel.
3. Skeletal pixels must be connected to each other to form several regions and the number of those regions should be equal to the number of regions in the original image.

5 Bidirectional Associative Memory

In 1988 Bart Kosko proposed an artificial neural network called bidirectional associative memory (BAM) [15]. BAM is a hetero-associative and content-addressable memory. A BAM network consists of two bipolar binary layers of neurons, say A and B. The neurons in the first layer (A) are fully interconnected to the neurons in the second layer (B). There is no interconnection between neurons in the same layer. Because the BAM network processes information in time and it involves bidirectional data flow, it differs in principle from a linear association, although both networks are used to store association pairs [16]. A general diagram of BAM is shown in Figure 3.

BAM algorithm [16]:

Step 1: The associations between pattern pairs are stored in the memory in the form of bipolar binary vectors with entries -1 and 1.

$$\{(a^{(1)}, b^{(1)}), (a^{(2)}, b^{(2)}), \ldots, (a^{(p)}, b^{(p)})\} \quad (1)$$

Vector a stores a pattern and is n-dimensional, while vector b is m-dimensional and stores the associated output.

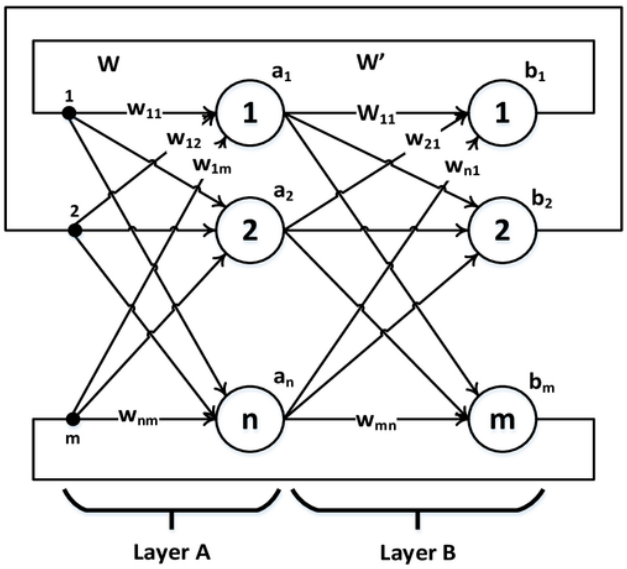

Figure 3 Bidirectional Associative Memory: General Diagram.

Step 2: Calculate weight using Eq. (2).

$$W = \sum_{i=1}^{p} a^{(i)}\, b^{(i)} t \tag{2}$$

Step 3: The test vector pair a and b is given as input.
Step 4: In the pass forward, b is given as input. Calculate a using Eq. (3).

$$a = \Gamma[Wb] \tag{3}$$

Calculate each element of vector a using Eq. (4).

$$a_i' = sgn\left(\sum_{j=1}^{m} w_{ij} b_j\right), for\ i = 1, 2, \ldots, n \tag{4}$$

Step 5: Vector a is now given as input to the second layer during the backward pass. Calculate the output of this layer using Eq. (5).

$$b' = \Gamma[Wa] \tag{5}$$

Calculate each element of vector b using Eq. (6).

$$b_j' = sgn\left(\sum_{i=1}^{n} w_{ij} a'_i\right), for\ j = 1, 2, \ldots, m \quad (6)$$

Step 6: Stop the process if there is no further update. Otherwise repeat step 4 and 5.

BAM storage capacity:

The maximum number of pattern pairs that can be stored and successfully retrieved is min (m, n). This estimate is heuristic. The memory storage capacity of BAM [16] is

$$P \leq min\ (m, n) \quad (7)$$

6 Counterpropagation Network

In 1987, Robert Hecht-Nielsen defined the counterpropagation network (CPN). The CPN is widely used because of its simplicity and ease of the training process. It also has good stats in the representation of the input layer for a wide range of environments. It combines an unsupervised training method on the Kohonen layer and a supervised training method on the Grossberg layer [17].

6.1 Forward Only Counterpropagation

The training for forward-only counterpropagation is the same as training for full counterpropagation. It consists of two phases; the first phase should be completed before proceeding to the second phase. The first phase is Kohonen learning and the second phase is Grossberg learning. In Figure 4 we can see the forward-only counterpropagation network architecture.

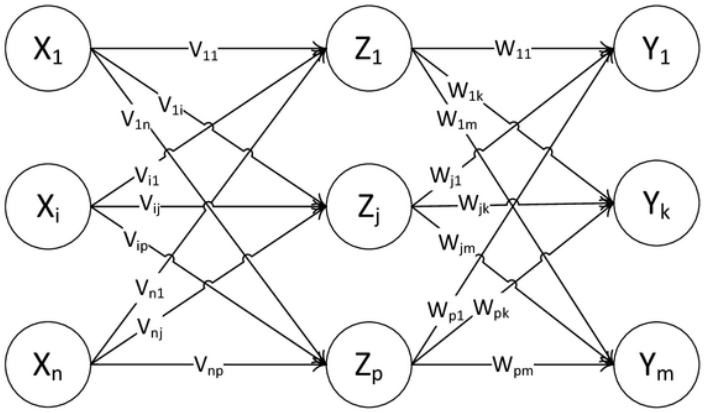

Figure 4 Forward only counterpropagation architecture.

Only the winner unit is allowed to learn or update the weights in the learning phase of the Kohonen training. The minimum distance between the weight vector and the input vector is calculated to determine the winning unit. Either Eqs. (8) or (9) can be used to calculate the distance between two vectors:

$$Dot\ Product\ (z\_inj = \sum xi\ vij + \sum yk\ wkj) \quad (8)$$

$$Euclidean\ Distance\ (Dj = \sum (xi - vij)2 + \sum (yk - wkj)2\ ) \quad (9)$$

When using dot product, look for results that have the largest value, because the larger the value of dot product, the smaller the angle between two vectors will become, provided that both vectors are normalized. When using Euclidean distance, look for results that have the smallest value, because Euclidean distance calculates the physical distance between the two vectors.

The training algorithm for forward-only counterpropagation is as follows [18]:

Step 0 : Initialize learning rate and all weights.
Step 1 : Do step 2-7 as long as the stop condition of the first phase is not met.
Step 2 : Do step 3-5 for each pair of input *x:y*.
Step 3 : Enter input vector *x* in input layer *X*.
Step 4 : Find the winner of the Kohonen layer unit, save its index into variable *J*.
Step 5 : Update weight for unit Z_j using Eq. 10

$$vijnew = (1 - \alpha)vijold + \alpha xi\ ,\ i = 1 \ldots n \quad (10)$$

Step 6 : Decrease the learning rate (α).
Step 7 : Check if the stop condition is met for the first phase.
Step 8 : Do step 9-15 as long as the stop condition of the second phase has not been met (α and β are very small and constant during the second phase).
Step 9 : Do step 10-13 for each pair of input *x:y*.
Step 10 : Enter input vector *x* in input layer *X*.
Step 11 : Find the winner of the Kohonen layer unit, save its index into variable *J*.
Step 12 : Update the weights that go into *Zj* using Eq. (11).

$$vijnew = (1 - \alpha)vijold + \alpha xi\ ,\ i = 1 \ldots n \quad (11)$$

Step 13 : Update the weights from Zj to the output layer using Eq. (12).

$$wjknew = (1 - \alpha)wjkold + \alpha yk\ ,\ k = 1 \ldots m \quad (12)$$

Step 14 : Decrease the learning rate (α).
Step 15 : Check if the stop condition is met for the second phase.

After the training process has been completed, the forward-only counterpropagation network can be used to map x into y using the following algorithm [18].

Step 0 : Use the weight training results.
Step 1 : Enter input vector x in input layer X.
Step 2 : Find the winner index unit, save in J.
Step 3 : Calculate output using Eq. (13).

$$Yk = wjk \tag{13}$$

7 Evolutionary Neural Network

An evolutionary neural network (ENN) is a combination of a neural network with an evolutionary algorithm. A common limitation of neural networks is associated with network training. Backpropagation learning algorithms have serious drawbacks, which cannot guarantee that the optimal solution is given. Another difficulty in neural network implementation is related to selecting the optimal network topology. Network architecture that is appropriate for certain cases are often chosen using heuristic methods. This shortcoming can be addressed using an evolutionary algorithm.

Evolutionary algorithms refer to a probabilistic adaptation algorithm inspired by natural evolution. This method follows the statistical search strategies in a population of individuals, each representing a possible solution to the problem. Evolutionary algorithms can divided into three main forms, namely evolution strategies, genetic algorithms, and evolutionary programming [19].

The evolutionary algorithm used in this research is a genetic algorithm. The genetic algorithm is an effective optimization technique that can help both optimizing the weight and selecting the network topology. A problem must be represented as a chromosome in order to use a genetic algorithm. When we want to look for a set of optimal weights of a multilayer feed-forward neural network, the first step in solving this problem is to have the system encode the network into a chromosome. This process can be seen in Figure 5 [20].

The second step is to define the fitness function in order to evaluate the performance of the chromosome. This function must be calculated given the performance of the neural network. A simple function from squared errors can be implemented. Each chromosome weight is given for each link in the network evaluate the fitness of the chromosomes. Collections of training examples are then presented to the network and the number of squared errors is calculated.

The genetic algorithm seeks to find the set amount of weight that has the smallest squared errors.

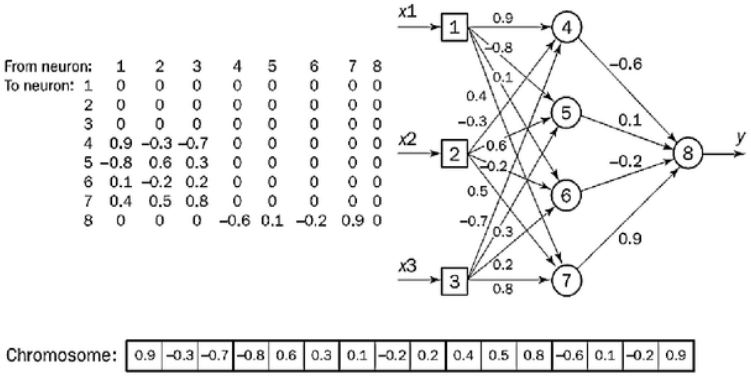

Figure 5 Encoding a network into a chromosome.

The third step is to choose the genetic operators crossover and mutation. The crossover operator requires two parent chromosomes and creates a child with genetic material from both of its parents. Each gene of the child chromosome is represented by the corresponding genes of a randomly selected parent. The system is ready to apply the genetic algorithm when the mutation operator randomly selects a gene and replaces it with a random result between -1 and 1. Users need to define the number of networks with different weights, the probability of crossover and mutation, the population number and the number of generations [20].

8 ICZ-ZCZ

Image centroid and zone (ICZ) and centroid zone and zone (ZCZ) are feature extraction methods that utilize the type of zoning and zone centroid of the zones that an image has been divided into. These methods begin with dividing an image into several equal zones.

After dividing the image into equal zones, the ICZ method calculates the centroid of the image. For each zone, the average distance between black image pixels and the centroid zone is calculated. In the ZCZ method, the centroid of the image is calculated instead of the centroid of each zone. And again, for each zone, the average distance between black image pixels and the image centroid is calculated. The average distances are then used as features for classification and recognition [21].

9 Backpropagation

A backpropagation neural network is a neural network that uses a multilayer feed-forward architecture. This method is widely used to solve many problems, such as classification, pattern recognition and generalization [22].

The training algorithms in backpropagation are as follows [23]:

Feed-forward phase (7 steps):

Step 0 : Initialize weight (random value between 0-1) and learning rate α
Step 1 : Do step 2-13 as long as the stop condition is not met
Step 2 : Perform steps 3-13 as the desired amount of training
Step 3 : Do steps 4-13 for each hidden layer and output layer
Step 4 : Calculate the input of each node in the hidden layer using Eq. (14).

$$z\_in_j = \sum_{i=1}^{n} x_i * w_{ij} \tag{14}$$

Step 5 : Calculate the output of each node in the hidden layer activation function using Eqs. (15) and (16)

$$z_j = f(z\_in_j) \tag{15}$$

$$f_1(x) = \frac{1}{1+\exp(-x)} \tag{16}$$

Step 6 : Calculate the input of each node in the output layer using Eq. (17)

$$y\_in_k = \sum_{j=1}^{n} z_j * w_{jk} \tag{17}$$

Step 7 : Calculate the output at each node in the output layer using Eq. (18)

$$y_k = f(y\_in_k) \tag{18}$$

Error-backpropagation phase (6 steps):

Step 8 : Calculate the error of each node in the output layer with the deactivation function using Eqs. (19) and (20)

$$\delta_k = (t_k - y_k) * f'(y\_in_k) \tag{19}$$

$$f_1'(x) = f_1(x)[1 - f_1(x)] \tag{20}$$

Step 9 : Calculate the change in weight for each output node in each layer using Eq. (21).

$$\Delta w_{jk} = \alpha * \delta_k \tag{21}$$

Step 10 : Calculate the error for each node in the hidden layer to deactivate the function using Eq. (22)

$$\delta_j = (\sum_{k=1}^{n} \delta_k * w_{jk}) * f'(z\_in_k) \tag{22}$$

Step 11 : Calculate the change in weight for each node in each hidden layer using Eq. (23)

$$\Delta w_{ij} = \alpha * \delta_j \tag{23}$$

Step 12 : Update the weight for each node in the output layer using Eq. (24)

$$w_{jk}(new) = w_{jk}(old) \qquad w_{jk} \tag{24}$$

Step 13 : Update the weight for each node in each hidden layer using Eq. (25)

$$w_{jk}(new) = w_{jk}(old) \qquad w_{jk} \tag{25}$$

10 Chi2

The Chi2 algorithm [24] is an algorithm that uses the $\chi 2$ statistic to discretize numeric valued attributes. This algorithm is quite effective if used in the selection of the important features of a group of numerical attributes. By using the features that are relevant, this algorithm can speed up the training process and improve the prediction accuracy of classification algorithms in general. Additionally, there are many classification algorithms that require and work better on discrete training data.

In use, the Chi2 algorithm is divided into two phases. The first phase begins with a high enough significance value, e.g. 0.5, for all attributes for discretization. The process of merging the data will continue for as long as $\chi 2$ does not exceed the specified significance value (0.5, yielding a value 0.455 with degree of freedom equal to 1). This phase will be repeated, reducing the significance value until the number of inconsistent data in the discretization exceeds the specified limit. The equation to calculate the value of $\chi 2$ can be seen in Eq. (26).

$$x^2 = \sum_{i=1}^{2} \sum_{j=1}^{k} \frac{(A_{ij} - E_{ij})^2}{E_{ij}} \tag{26}$$

k = number of classification,

A_{ij} = number of pattern at interval - i, classification - j

E_{ij} = the pattern expected from Aij = $R_i * C_j / N$, if R_i or C_j equal to 0,

E_{ij} should change to 0.1

R_i = number of pattern at interval - i = $\sum_{j=1}^{k} A_{ij}$

C_j = number of pattern at interval - j = $\sum_{i=1}^{2} A_{ij}$

N = total number of pattern = $\sum_{i=1}^{2} R_i$

The second phase is an optimization of the first phase. The most visible difference is the calculation of inconsistency. Calculation is done after all the attributes have gone through the merger process in the second phase. The inconsistency value is calculated at the end of each attribute discretization while in the first phase. The second phase will be repeated until there are no longer any values of attributes that can be discretized or combined. Inconsistency occurs when there are several samples with all of their attributes having the same value but they belong to different groups.

11 Design and Implementation

The input for the system is a Javanese character digital image. Grayscale processing and filtering are done to reduce noise. Subsequently, we apply skew detection and correction to straighten skewed images. Later, the segmentation process is executed to get each Javanese character using thresholding and skeletonizing. Feature extraction is done using ICZ-ZCZ [21] and the features will be used as inputs for the artificial neural network.

Each Javanese character image is divided into 4*5 zones, after which ICZ-ZCZ will be performed for each zone, so there are 40 ICZ-ZCZ output values. These values will later become artificial neural network input nodes.

The overall system workflow can be seen in Figure 6.

The application interface is shown in Figure 7.

Having obtained the image of each Javanese character from the document and having carried out the feature extraction process on each image, the next step performed by the system is to identify the characters using the artificial neural network methods. After successful recognition, the Javanese character images are converted into a textual sequence in a *Hanacaraka* font and formed into a document.

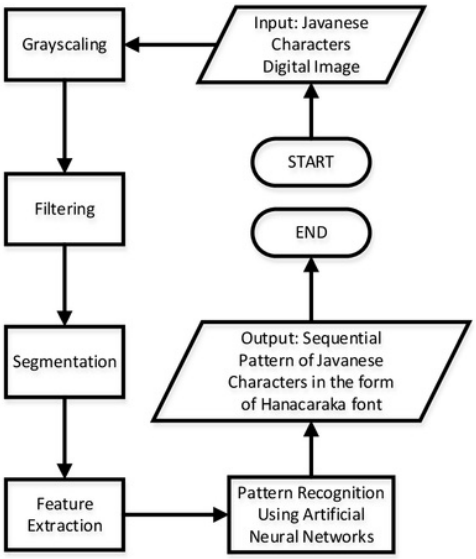

Figure 6 System workflow.

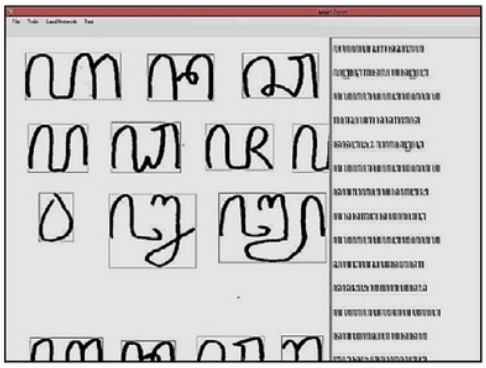

Figure 7 Application interface.

11.1 Dataset for the Experiment

For the experiment, we used two kinds of datasets of handwritten Javanese characters, one dataset for training and the other set for testing. For training CPN, BPNN and ENN, the number of data in the dataset was 20 samples for

each character. There are in total 31 Javanese characters, so the overall number of data was 620 characters. The dataset for testing also consisted of 20 sample data for each Javanese character, with the overall data being 620 characters. Examples of the sample data can be seen in Figure 8. Every sample data was processed by feature extraction using the ICZ-ZCZ method. The 40 value results of the feature extraction became the inputs for 40 nodes of the neural network input neurons.

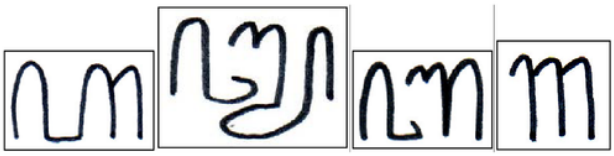

Figure 8 Examples of data samples for training and testing process.

12 Experimental Results

For pattern recognition of Javanese characters, we used five kinds of artificial neural networks, namely bidirectional associative memory, counterpropagation, evolutionary, backpropagation and backpropagation combined with chi2.

The experimental results of bidirectional associative memory (BAM) can be seen in Table 2.

Table 2 Experimental Results of BAM.

Number of sample	Input node	Output node	Accuracy (%)
	6	4	100.00
2	15	10	0.00
	30	10	0.00
	6	4	100.00
3	15	10	33.33
	30	10	0.00
	6	4	100.00
4	15	10	0.00
	30	10	0.00
	6	4	66.67
6	15	10	0.00
	30	10	0.00
	6	4	75.00
8	15	10	0.00
	30	10	0.00

From the experimental results above, we can see that BAM was inaccurate when applied to Javanese character recognition. This is because we needed at least 40 nodes for the input, while BAM only works well when the number of inputs is the same as or less than 6 nodes. For the output we needed 31 nodes in this experiment because the total number Javanese characters is 31, while BAM only works well for 4 nodes or less.

Another experiment used a counterpropagation network (CPN), a backpropagation network (BPNN) and an evolutionary neural network (ENN) with 1 layer and 2 layers. We performed the experiment using a dataset that had been trained beforehand and a dataset that had not been trained beforehand.

The neural network output layer consisted of 31 neurons. These 31 neurons were in accordance with the number of Javanese characters used in this research: 20 basic (*carakan*) characters, 4 *sandhangan* characters, and 7 number characters (not all 10 number characters, because 3 number characters have the same form as *carakan* characters). Each neuron has a value of 0 or 1. For the first character, the first neuron is 1, while the other neurons are 0. For the second character, the second neuron is 1, while the other neurons are 1, etc. The number of neurons in the other layer is 60.

From the experimental results it can be seen that the average recognition accuracy of CPN was only about 71% for trained data and 6% for test data (data had not been trained beforehand). The average recognition accuracy of ENN was about 94% for trained data and about 66% for test data. The parameters used for ENN were: the number of neurons for each layer: 60, crossover probability: 100%, mutation probability: 50%, maximum population: 50, maximum epoch: 10 million, and error limit: 0.1. The average recognition accuracy of BPNN was about 79% for trained data and 33% for test data. The parameter used for BPNN: input neurons: 40, learning rate: 0.1, error threshold: 0.001.

We tried to improve the accuracy of recognition by using an additional method, Chi2, and combining it with a backpropagation neural network. The Chi2 method was used to further increase the variation of the features of each data to be trained or recognized. With more data features it was expected that the differences in the features of each Javanese character would be accentuated. The experiment with a combination of Chi2 and BPNN was done using the following parameters: feed-forward network with one hidden layer, number of neurons in hidden layer: 60, maximum epoch: 1000, learning rate: 0.1, initial input neuron: 40 features combined with output of Chi2 algorithm to become 440 input neurons (using N=10, for each input will produce 10 outputs), output neurons: 31. The overall experimental results can be seen in Table 3 for the

experiment using data that had been trained beforehand and Table 4 for the experiment using test data.

Table 3 Experimental results using trained data.

Javanese characters type	Recognition accuracy (%)				
	CPN	BPNN	ENN 1 layer	ENN 2 layers	Chi2 and BPNN
Basic character / *Carakan*	61.25	66.25	97.75	96.00	98.25
Numbers	73.57	72.14	97.14	97.86	97.86
Sandhangan	77.50	78.75	93.75	90.00	97.50
All characters type	71.45	79.03	94.19	92.26	98.71

Table 4 Experimental results using test data.

Javanese characters type	Recognition accuracy (%)				
	CPN	BPNN	ENN 1 layer	ENN 2 layers	Chi2 and BPNN
Basic character / *Carakan*	4.75	32.25	48.75	52.75	65.75
Numbers	6.43	32.14	58.57	63.57	79.29
Sandhangan	7.50	36.25	66.25	68.75	83.75
All characters type	6.29	33.87	50.32	66.29	73.71

13 Conclusion

In this research, we have developed a handwritten Javanese character recognition system using several artificial neural network methods and compared their recognition results. From the experiment that has been executed it can be concluded that the bidirectional associative memory method and the counterpropagation network method cannot be used for recognition of Javanese characters because their average accuracy was very low. The combination of the Chi2 method and the backpropagation neural network method performed better than the evolutionary neural network method with 1 layer or 2 layers for Javanese character recognition. Its recognition accuracy rate reached 98% for data that had been trained beforehand and 73% for data that had not been trained beforehand. For future research, the accuracy rate may be improved by using another method for segmentation and feature extraction that can better distinguish similar Javanese characters. Also a combination of Chi2 and ENN may be used to improve accuracy.

Acknowledgments

This research was funded by Research Competitive Grant DIPA-PT Coordination of Private Higher Education Region VII, East Java, Indonesia (20/SP2H/PDSTRL_PEN/LPPM-UKP/IV/2014), fiscal year 2014 and 2015.

This research was also funded by Research Center, Petra Christian University, Surabaya, Indonesia, through the Internal Research Grant (05/Pen-LPPM/UKP/2012), fiscal year 2012. We also thank Edwin Prasetio Nandra, Danny Setiawan Putra, Eric Yogi Tjandra, Evan Sanjaya, Jeffry Hartanto, Ricky Fajar Adi Edna P., and Christopher H. Imantaka for their help in doing the system coding.

References

[1] Nurmila, N., Sugiharto, A. & Sarwoko, E.A., *Back Propagation Neural Network Algorithm for Java Character Pattern Recognition*, Jurnal Masyarakat Informatika, **1**(1), pp. 1-10, 2010.

[2] Priyatma, J.E. & Wahyuningrum, S.E., *Java Character Recognition Using Fuzzy Logic*, SIGMA, **8**(1), pp. 75-84, 2005.

[3] Adipranata, R., Budhi, G.S. & Thedjakusuma, R., *Java Characters Word Processing*, in Proceeding of The 3rd International Conference on Soft Computing. Intelligent System and Information Technology, Bali, Indonesia, 2012.

[4] Budhi, G.S. & Adipranata, R., *Comparison of Bidirectional Associative Memory, Counterpropagation and Evolutionary Neural Network for Java Characters Recognition*, in Proceedings of The 1st International Conference on Advanced Informatics: Concepts, Theory and Applications, Bandung, Indonesia, 2014.

[5] Budhi, G.S. & Adipranata, R., *Java Characters Recognition using Evolutionary Neural Network and Combination of Chi2 and Backpropagation Neural Network*, International Journal of Applied Engineering Research, **9**(22), pp. 18025-18036, 2014.

[6] Adipranata, R., Budhi, G.S. & Setiahadi, B., *Automatic Classification of Sunspot Groups for Space Weather Analisys*, International Journal of Multimedia and Ubiquitous Engineering, **8**(3), pp. 41-54, 2013.

[7] Budhi, G.S., Adipranata, R. & Jimmy Hartono, F., *The Use of Gabor Filter and Backpropagation Neural Network for Automobile Types Recognition*, in Proceeding of The 2nd International Conference on Soft Computing, Intelligent System and Information Technology, Bali, Indonesia, 2010.

[8] Budhi, G.S., Purnomo, M.H. & Pramono, M., *Recognition of Pancreatic Organ Abnormal Changes Through Iris Eyes Using Feed-Forward Backpropagation Artificial Neural Network,* in Proceeding of The 7th National Conference on Design and Application of Technology, 2008.

[9] Budhi, G.S., Gunawan, I. & Jaowry S., *Backpropagation Artificial Neural Network Method for Recognition of Characters on Digital Image*, in Proceeding of the 3rd National Conference on Design and Application of Technology, 2004.

[10] *Java Character Hanacaraka*, http://nusantaranger.com/referensi/buku-elang/chapter-4merah/aksara-jawa-hanacaraka/, last access October 2014.
[11] *Javanese Alphabet* (*Carakan*), http://www.omniglot.com/writing/javanese.htm (October 2014).
[12] *Java Characters*, http://id.wikipedia.org/wiki/Aksara_Jawa (January 2013).
[13] Gonzalez, R.C. & Woods, R.E., *Digital Image Processing*, 3rd Edition, New Jersey: Prentice-Hall, Inc., 2008.
[14] Parker, J.R., *Algorithm for Image Processing and Computer Vision*, New York: John Wiley and Sons, Inc., 2010.
[15] Kosko, B., *Bidirectional Associative Memories*, IEEE Transactions on Systems, Man, and Cybernetics, **18**(1), pp. 49-60, 1988.
[16] Singh, Y.P., Yadav, V.S., Gupta, A. & Khare A., *Bi Directional Associative Memory Neural Network Method In The Character Recognition*, Journal of Theoretical and Applied Information Technology, **5**(4), pp. 382-386, 2009.
[17] Boyu, W., Feng W. & Lianjie S., *A Modified Counter-Propagation Network for Process Mean Shift Identification*, in Proceeding of IEEE International Conference on Systems, Man and Cybernetics, pp. 3618-3623, 2008.
[18] Fu, L.M., *Neural Networks in Computer Intelligence,* New York: McGraw-Hill, Inc, 1994.
[19] Dewri, R., *Evolutionary Neural Networks: Design Methodologies*, http://ai-depot.com/articles/evolutionary-neural-networks-design-methodologies/ (January 2013).
[20] Negnevitsky, M, *Artificial Intelligence: A Guide to Intelligence Systems* (2nd ed.), New York: Addison Wesley, 2005.
[21] Rajashekararadhya, S.V. & Ranjan, P.V., *Efficient Zone Based Feature Extraction Algorithm for Handwritten Numeral Recognition of Four Popular South Indian Scripts*, Journal of Theoretical and Applied Information Technology, **4**(12), pp. 1171-1181, 2005.
[22] Rao, H.V. & Valluru B.R., *C++ Neural Networks and Fuzzy Logic,* New York: Henry Holt and Company, 1993.
[23] Fausett, L., *Fundamentals of Neural Networks,* New Jersey: Prentice Hall, 1994.
[24] Liu, H. & Setiono, R., *Chi2: Feature Selection and Discretization of Numeric Attributes*, In Proceeding of The 7th International Conference on Tools with Artificial Intelligence, pp. 388-391, 1995.

Handwritten Javanese Character Recognition Using Several Artificial Neural Network Method

ORIGINALITY REPORT

11%	9%	11%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	emitter.pens.ac.id Internet Source	3%
2	www.jatit.org Internet Source	2%
3	www.scribd.com Internet Source	1%
4	www.csse.melbourne.edu Internet Source	1%
5	Lecture Notes in Computer Science, 2007. Publication	1%
6	Communications in Computer and Information Science, 2015. Publication	1%
7	lecturer.polindra.ac.id Internet Source	1%
8	Submitted to School of Business and Management ITB Student Paper	1%

9 Submitted to University of East London
Student Paper 1%

Exclude quotes	On	Exclude matches	< 1%
Exclude bibliography	On		