Comparison of Bidirectional Associative Memory, Counterpropagation and Evolutionary Neural Network for Java Characters Recognition

Gregorius Satia Budhi
Informatics Department
Petra Christian University
Surabaya, Indonesia
greg@petra.ac.id

Rudy Adipranata
Informatics Department
Petra Christian University
Surabaya, Indonesia
rudya@petra.ac.id

***Abstract*— Javanese language is the language used by the people on the island of Java and it has its own form of letters called Java characters. Recognition of Java characters is quite difficult because it consist of basic characters, numbers, complementary characters, and so on. In this research we developed a system to recognize Java characters and compared three methods of neural network namely bidirectional associative memory, counterpropagation and evolutionary neural network. Input for the system is a digital image containing several Java characters. Digital image processing and segmentation are performed on the input image to get each Java character. For each Java character, feature extraction is done using ICZ-ZCZ method. Output from feature extraction will become input for neural network. From experimental result, evolutionary neural network can perform better recognition accuracy than the other two methods.**

***Keywords*—Java characters recognition, bidirectional associative memory, counterpropagation, evolutionary neural network**

I. INTRODUCTION

Javanese language is the language used by the people on the island of Java. Javanese language has its own form of letters referred to the character of Java. Java characters recognition has its own difficulty level because of basis characters, vowels, complementary, and so on. Because it is difficult to recognize, then lately not many people can do the writing or reading of Java characters. For many people, Java characters eventually regarded as a decoration only and do not mean anything. This will further erode gradually the existence of Java characters and will ultimately also affect the Javanese culture in general.

Some researchers have conducted research on this Java character recognition. Nurmila [1] used backpropagation neural network and the accuracy result was about 61%. Other researcher, Priyatma used fuzzy logic for recognition [2] and the recognition results are satisfactory.

In this research, we developed a system that can automatically recognize Java characters in the form of digital image, and turn them into a document written with the *hanacaraka* font. The first process is digital image segmentation and feature extraction. The features will be used as input for neural network. In this research we compare three methods of neural networks, namely bidirectional associative memory, counterpropagation network, and evolutionary neural network.

II. JAVA CHARACTERS

Java characters are differ from the commonly used Latin characters. Java characters have different shape and structure with the Latin characters. *Carakan* characters is the core of Java characters consisting of 20 syllables called Dentawyanjana, can be seen in Figure 1 [3].

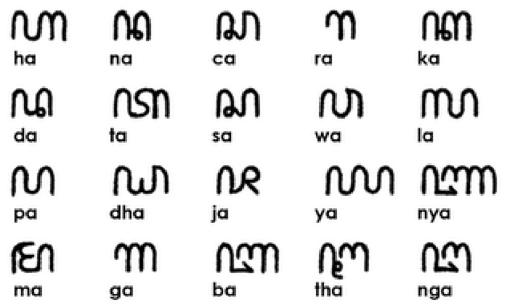


Fig. 1. Basic (*carakan*) characters

For numbers in Java characters can be seen in Figure 2 [3].

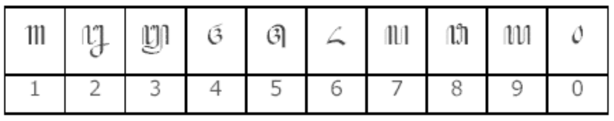
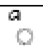

1
2
3
4
5
6
7
8
9
0

Fig. 2. Symbol of numbers

Sandhangan character is commonly used as complementary character, vowel or consonant that are commonly used in everyday language. *Sandhangan* can be seen in Figure 3 [3].

Sandhangan name	Java character	Description
Wulu		Vowel i

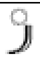
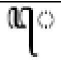
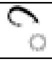
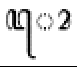
Suku		Vowel u
Taling		Vowel é
Pepet		Vowel ê
Taling tarung		Vowel o

Fig. 3. *Sandhangan* characters

III. Image Segmentation

Segmentation is one of the important processes used to transform the input image to the output image taken based on the attributes of the image. Segmentation divides the image into regions based on its intensity so can distinguish objects and background. Segmentation should be discontinued if each object has been isolated or clearly visible [4]. The segmentation methods that used in this research are thresholding and skeletonizing.

Thresholding is one way to separate the objects in the image from the background by selecting a threshold value T that can separate these two modes. With the election of the value of T, all points (x, y) where f (x, y)> T, can be called an object point and besides it is called a background point or vice versa [4].

Skeletonizing or thinning is the process to get rid of the extra pixels and produces images that are more modest. The purpose of skeletonizing is made simpler image so that the image can be analyzed further in the way of its shape and suitability. Problem encountered in conducting thinning is how to determine the pixels are redundant. If we cannot determine it, the thinning process is more likely to an erosion process where erosion can cause a region is deleted. Skeleton should remain intact and have some basic properties such as [5]:

- Must consist of several thin regions, with a width of 1 pixel.
- Pixels that form the skeleton should be near the middle are of the cross section of the region.
- Skeletal pixel must be connected to each other to form several regions that are equal to the number of region in original image.

IV. Bidirectional Associative Memory

Bidirectional associative memory (BAM) proposed by Bart Kosko in 1998 [6]. This method associates the patterns of a set, for example set A to set B, the group or other set of patterns, and vice versa. BAM architecture can be seen in Figure 4.

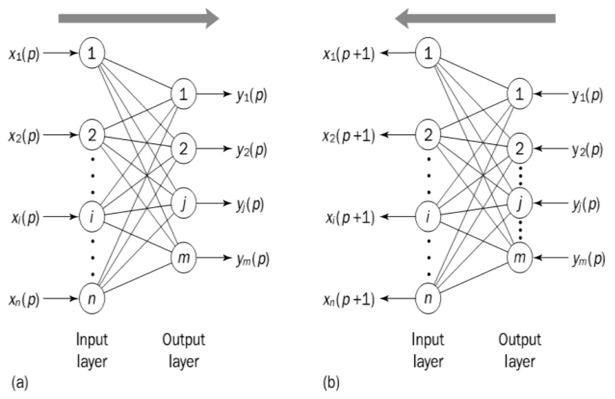


Fig. 4. Architecture of BAM: (a) forward direction (X → Y); (b) backward direction (Y → X)

V. Counterpropagation Network

Counterpropagation network (CPN) is defined by Robert Hecht-Nielsen in 1987 [7]. This method is widely used because it is simple and easy on the training process. Additionally CPN has good stats in the representation of the input layer for a wide range of environment. CPN combines unsupervised training method on Kohonen Layer and supervised on Grossberg layer [7]. Network topology of CPN can be seen in Figure 5.

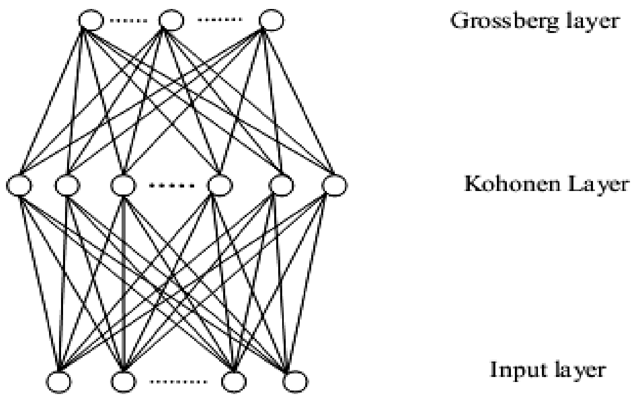


Fig. 5. Counterpropagation network topology

VI. Evolutionary Neural Network

Evolutionary neural network (ENN) is a combination of a neural network with evolutionary algorithm. Although the neural network can be used to solve various kinds of problems, it still has some limitations. A common limitation is usually associated with network training. Backpropagation learning algorithms are often used as flexible and easy to implement had serious drawbacks, which cannot guarantee that the optimal solution is given. Another difficulty is related to selecting the optimal network topology for the neural network. Network architecture that is appropriate for certain cases more often chosen from heuristic methods, and neural network topology design is still an art than a technique. This shortcoming can be addressed using evolutionary algorithm.

Evolutionary algorithm refers to a probabilistic adaptation algorithm inspired from natural evolution. This method follows the statistical search strategies in a population of individuals, each representing a possible solution to the problem. Evolutionary algorithm divides into three main forms, namely: evolution strategies, genetic algorithms, and evolutionary programming [8].

In this research, the evolutionary algorithm used is the genetic algorithm. Genetic algorithm is an effective optimization technique that could help both the optimization of weight and selecting the network topology. In order to use genetic algorithm, first a problem must be represented as a chromosome. For example, when we want to look for a set of optimal weight of a multilayer feed forward neural network, the first step in solving this problem is the system should make the process of encoding of the network into a chromosome as in Figure 6 [6].

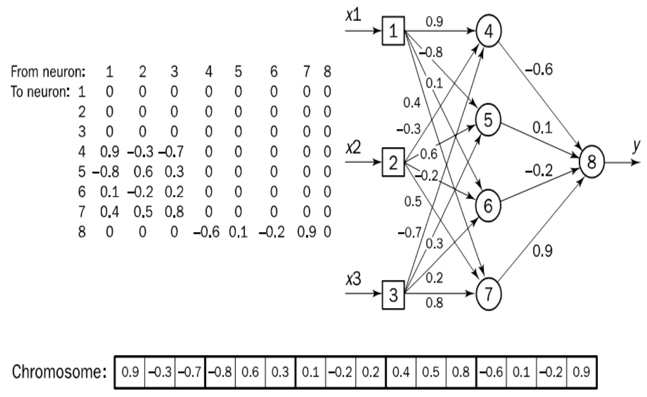


Fig. 6. Encoding a network into a chromosome

The second step is to define the fitness function to evaluate the performance of the chromosome. This function must be calculated given the performance of the neural network. We can implement a simple function from squared errors. To evaluate the fitness of the chromosomes, each chromosome weight is given to each link in the network. Training of examples collections are then presented to the network, and the number of squared errors is calculated. Small squared errors indicate that the chromosome is more fit than the other. In other words, genetic algorithm seeks to find a set amount of weight that has the smallest squared errors.

The third step is to choose the genetic operators, namely crossover and mutation. Crossover operator requires two parent chromosomes and creates a child with genetic material from both of its parent. Each gene of the child chromosome is represented by the corresponding genes of randomly selected parent. Mutation operator randomly selects a gene and replaces it with a random result between -1 to 1. By doing so, the system is ready to apply genetic algorithms. However, users still need to define the number of population, the number of networks with different weights, the probability of crossover and mutation as well as the number of generation [6].

VII. Implementation and Result

System workflow starting from input of a Java characters digital image. Then we do the grayscale processing and filtering to remove noise that exists. After that the segmentation process is carried out to get the parts of hanacaraka character using thresholding and skeletonizing. Later feature extraction process is done by using ICZ-ZCZ [9] and the feature will be used as inputs to the neural network.

ICZ (Image Centroid and Zone) – ZCZ (Zone Centroid and Zone) is zoning type feature extraction that utilizing centroid of the image or centroid of the zone. Each digital image input (each Java character image) is divided into 20 zones (4 * 5 zones), and for each zone, the ICZ and ZCZ methods will be performed so there are 40 ICZ-ZCZ output values that become neural network input node.

The overall system workflow can be seen in Figure 7.

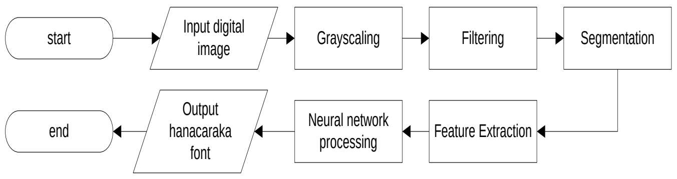


Fig. 7. System workflow

Application interface can be seen in Figure 8.

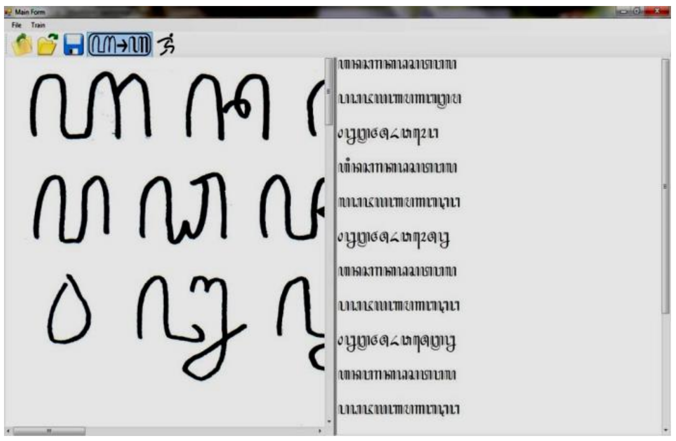


Fig. 8. Application interface

Experimental results of bidirectional associative memory (BAM) can be seen in Table 1.

TABLE I. Experimental Result of BAM

No	Number of sample	Input node	Output node	Accuracy (%)
1	2	6	4	100.00%
2	2	15	10	0.00%
3	3	4	5	100.00%
4	4	6	3	100.00%
5	6	6	3	66.67%
6	6	6	3	33.33%
7	6	6	4	100.00%
8	8	6	4	75.00%
9	8	6	5	62.50%
10	8	6	5	75.00%

11	8	8	5	37.50%
12	4	3	1	0.00%
13	3	30	10	33.33%
14	4	30	15	0.00%
15	4	30	30	0.00%

From the experimental results above it can be concluded that the BAM is inaccurate to use for Java characters recognition. For input node, we need at least 40 nodes, while BAM only works well when the input nodes are the same or less than 6 nodes only. And for the output nodes, we need at least 20 nodes because Java characters consists of at least 20 basic characters, not included numbers and *sandhangan*, while BAM works well for 3 or 4 nodes only.

Another experiments use counterpropagation network (CPN) and evolutionary neural network (ENN) 1 layer and 2 layers, and from experimental result, the average of recognition accuracy of CPN is only about 70% for training data and 4% for testing data, while the average of recognition accuracy of ENN is about 94% for training data and about 62% for testing data. Parameters used for ENN are: the number of neuron for each layer: 60, crossover probability: 100%, mutation probability: 50%, maximum population: 50, maximum epoch: 10 million and error limit: 0.1. The experimental result of CPN and ENN can be seen in table II.

TABLE II. EXPERIMENTAL RESULT OF CPN AND ENN

Character type	Data type	Accuracy (%)		
		CPN	ENN (1 layer)	ENN (2 layers)
All characters (basic / *carakan*, number & *sandhangan*)	Training	70.22	94.90	93.53
	Testing	4.76	58.23	62.38
Basic / *carakan*	Training	60.28	97.67	96.33
	Testing	3.17	58.12	59.31
Numbers	Training	73.14	99.33	98.67
	Testing	5.02	60.84	64.85
Sandhangan	Training	77.20	93.33	88.89
	Testing	6.26	66.92	68.12

CONCLUSION

From the experimental that has been done, it can be concluded that bidirectional associative memory and counterpropagation neural network could not be used for recognition of Java characters because the average of accuracy is very low, while evolutionary neural network still could be used for Java characters recognition because from the experimental result, it show that the average of accuracy is quite high. For future research, the accuracy may be improved by using another method for feature extraction that can distinguish similar Java character.

ACKNOWLEDGMENT

We thank the Research Center, Petra Christian University, Surabaya, Indonesia, which has funded this research 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., dan Sarwoko, E. A., "Back Propagation Neural Network Algorithm For Java Character Pattern Recognition," Jurnal Masyarakat Informatika vol 1, no 1, pp 1-10, 2010.

[2] Priyatma, J. E. dan Wahyuningrum, S. E., "Java Character Recognition Using Fuzzy Logic," SIGMA vol 8, No 1, pp 75-84, 2005.

[3] Java Characters, Aksara Jawa, http://id.wikipedia.org/wiki/Aksara_Jawa, last access January 2013.

[4] Gonzalez, R.C., and Woods, R.E., "Digital Image Processing 3rd Edition," New Jersey: Prentice-Hall, Inc., 2008.

[5] Parker, J.R., "Algorithm for Image Processing and Computer Vision," New York: John Wiley and Sons, Inc., 2010.

[6] Negnevitsky, M, "Artificial Intelligence: A Guide to Intelligence Systems (2nd ed.)," New York: Addison Wesley, 2005.

[7] Boyu, W., Feng W., and Lianjie S., "A Modified Counter-Propagation Network for Process Mean Shift Identification," IEEE International Conference on Systems, Man and Cybernetics. pp. 3618 – 3623, 2008.

[8] Dewri, R., "Evolutionary Neural Networks: Design Methodologies,“ http://ai-depot.com/articles/evolutionary-neural-networks-design-methodologies/, last access January 2013.

[9] Rajashekararadhya, S.V., Ranjan, Vanaja, "Efficient zone based feature extraction algorithm for handwritten numeral recognition of four popular South Indian scripts," Journal of Theoritical and Applied Information Technology 4(12), pp. 1171-1181, 2005.