2 Detection of Grayscale Image Implementation Using Multilayer Perceptron

O N Reni¹ and E Marpanaji²

¹,²Electronics and Informatics Engineering Education, Yogyakarta State University, Indonesia

Email: ¹oktina.reni@gmail.com; ²eko@uny.ac.id

Abstract. The human eye can instantly recognize two patterns of color images that are almost the same quickly, but the computer cannot directly recognize any pattern in the image. The problem faced is how the computer can recognize the image pattern entered. Pattern recognition is also a technique that aims to classify previously processed images based on similarity or similarity in characteristics. In the Artificial Neural Network there are several methods that can be used to identify image patterns, one of them with the Multilayer Perceptron architecture. Multilayer Perceptron Neural Network is a type of neural network that has the ability to detect or perform analysis for problems that are sufficient or even very complex, such as in language processing problems, recognition of a pattern and processing of an image or image. The results of the study are a system that is able to recognize grayscale image patterns and is able to provide a percentage of pattern recognition in two similar and different images.

Keywords: grayscale multilayer perceptron; neural network; backpropagation.

1. Introduction

The human eye can immediately recognize two colored images that are almost the same quickly, but the computer cannot immediately recognize the image. The problem faced is how the computer can recognize the inserted image.

Image Similarity Detection is almost similar to Pattern Recognition, which is a technique aimed at classifying previously processed images based on the similarity or similarity of the features they have. In Artificial Neural Networks, there are several methods that can be used to detect image similarities, one of which is the Multilayer Perceptron architecture. Multilayer Perceptron is a type of neural network that has the ability to detect or perform analysis for problems that are quite or even very complex, such as problems with language processing, similarity detection, and image processing.

In this research, the writer made a similarity detection program between grayscale and Multilayer Perceptron images. In this case, the file type to be used is a bitmap image (.bmp). Through the use of this Artificial Neural Network, especially the Backpropagation Network method, it is hoped that it will be implemented in detecting the similarity of the image grayscale.
Based on the background description above, the writer formulates the problem to be investigated, namely how can Artificial Neural Networks be used to detect similarities in grayscale images using Multilayer Perceptron?

**Research problem limitation:**
1. The method used for image similarity detection is using Artificial Neural Networks on the Multilayer Perceptron architecture.
2. Input / input given is an image with the file type .bmp (Bitmap) 8-bit, 16-bit, and 24-bit.
3. Image is limited to a maximum size (pixel) of 100x100.

In this research, a grayscale image similarity detection program will be made using the Multilayer Perceptron.

### 2. Research Methodology

Dhaneswara states that an Artificial Neural Network with a Multilayer Perceptron architecture and using back propagation training using momentum can be used as a data classification technique [3]. One of the main problems in ANN is the length of the training process (network model formation), therefore choosing the right network configuration (number of hidden layers, neurons, momentum value, learning-rate, activation function) is needed to speed up the training process. However, this configuration can differ from one set of training data to another, so experimentation is needed to find it.

Pattern recognition is a discipline that classifies objects based on image, weight or predetermined parameters into a number of categories or classes. Pattern recognition is also a technique that aims to classify previously processed images based on the similarity or similarity of features they have.

According to Renaldi Munir, Bitmap is a bit mapping, meaning that the pixel intensity value in the image is mapped to a certain number of bits [12]. A common bit map is 8, meaning that each pixel is 8 bits long. These eight bits represent the pixel intensity value. Thus there are as many as 256 degrees of gray, ranging from 0 to 255.

There are 3 types of bitmap formats, namely monochrome (binary image), grayscale and color. Grayscale image: there are only 2 colors. Each pixel only contains 1 bit (0 or 1) of information to represent the pixel value. The color seen is a combination of 3 basic colors, namely Red, Green, Blue (RGB).

In this study, the image to be used is a grayscale image. Grayscale image: in the form of 256 shades of gray. Each pixel contains 8-bit (0-255) color information. In a grayscale image, the value indicates that a gray image has only one color channel. The gray image is generally an 8 bit image.

Artificial Neural Network is a representation of the artificial human brain network in describing the learning process in the human brain. Artificial neural networks are implemented in computer programs in order to be able to complete several calculation processes during the learning process. The Artificial Neural Network consists of several neurons, and there are connections between neurons as in the human brain. Neurons change the information received through the output to other neurons. The relationship between the descendants is known as weights. The information is stored at a certain value in that weight.

Based on the connection patterns between neurons, there are 3 main characteristics of Artificial Neural Network systems that are often used, namely: Single Layer Neural Network, Recurrent Neural Network, and Multilayer Perceptron Neural Network.

In this study, the architecture to be used is the Multilayer Perceptron architecture. The Multilayer Perceptron model has additional neuron screens in addition to the input and output screens, which is a hidden layer located between the two screens. The number of hidden layers varies depending on the level of difficulty of the problems handled by the system, so that in applying Multilayer Perceptron it is more powerful than other Artificial Neural Network models. Figure 1 is a form of the Multilayer Perceptron network architecture:
Artificial Neural Networks have 3 learning methods, namely Unsupervised Learning, Semi Supervised Learning, and Supervised Learning. In this study using the Supervised Learning method, any knowledge that is given a reference value for mapping an input, will be the desired output. The learning process will be carried out continuously as long as the desired error conditions have not occurred. For each error value obtained at each learning stage, it will be calculated until the desired data or target value is achieved.

The Backpropagation training method is a type of supervised learning where the output of the network is compared with the expected target so that an output error is obtained with the delta rule, then this error is propagated back to update the network weight in order to minimize errors. Supervised learning algorithms are typically used by multi-layered perceptrons to change the weights associated with neurons in a hidden layer called Backpropagation. The Backpropagation algorithm uses an output error to change the backward weight value. The forward propagation stage must be carried out first in order to get an error. The sigmoid activation function is used to activate neurons during the forward propagation process. The requirements that must be met so that the activation function on Backpropagation is used are:

1. Continuous
2. Differential easily
3. Is an ascending function.

The binary sigmoid function with a range (0,1) is a function that fulfills the three requirements of the activation function in Backpropagation.

Definition of the binary sigmoid function:

\[ f(x) = \frac{1}{1 + e^{-x}}, -\infty \leq x \leq \infty \]  
\[ f'(x) = f(x)(1 - f(x)) \]  
\[ f(x) = \frac{2}{1 + e^{2x}} - 1, -\infty \leq x \leq \infty \]  
\[ f'(x) = \frac{2(1 - f(x))(1 - f(x))}{2} \]

The bipolar sigmoid function with range (-1,1) is defined as follows

\[ f(x) = \frac{2}{1 + e^{2x}} - 1, -\infty \leq x \leq \infty \]

The maximum value of the sigmoid function is 1. For patterns that have more than 1 target, the output and input patterns must be changed first so that all patterns have the same range as the sigmoid function. The activation function in the output layer uses the identity function \( f(x) = x \).

Training using the Backpropagation method consists of three steps, namely: [5]

a. feedforward
b. Calculation and backpropagation of the error in question
c. Updated weights and biases.

During the feedforward process the input unit \( x_i \) will receive the input signal and will send the signal to each hidden unit \( z_j \). The hidden unit will count the activation and send a signal \( z_j \) to the
output unit. Then, each unit of output will calculate the activation to produce a response to the input given by the network. During the training process, each unit of output compares the activation to the desired output value to determine the amount of error. From the existing errors, the factor (δk) is calculated. Factors are used to distribute the error from the output back to the previous layer. The factor (δi) is also calculated in hidden units (zj), as in the calculation of the factor (dk). Updating the weight between the hidden layer and the input layer using a factor (dk). The layer weights are adjusted together after all the factors are determined. The update of the weight (wjk) is carried out based on the result of the factor (δk) and the activation of the hidden unit. The update of weight (vij) is carried out based on the results of the factor (δj) and activation (xi) of the input.

The steps of back propagation training are as follows [5]:

Step 0. Initialize the weights
Step 1. When the stop condition is wrong, do steps 2 - 9
Step 2. For each training pair, do steps 3 - 8

Feedforward
Step 3. Each input unit (xi, i = 1 ...... n) receives the input signal (xi) and sends a signal to all units in the upper layer (hidden unit).
Step 4. Each hidden unit (zj, j = 1 ...... p) adds up the signal weights input,
\[ z_{in} = v_{0j} + \sum_{i} v_{ij}x_i \]  
apply the activation function to calculate the signal the output:
\[ z_j = f(z_{in}) \]  
and send signals to all units in the layer above it (unit output).
Step 5. Each unit of output (Yk, k = 1,..., M) adds a weight input signal.
\[ y_{in} = w_{0k} + \sum_{j} z_jw_{jk} \]  
and applies its activation function to compute output signal
\[ y_k = f(y_{in}) \]

Backpropagation:
Step 6. Each unit of output (Yk, k = 1, ..., m) receives a target pattern according to the input training pattern, calculate the error information.
\[ \delta_k = (t_k - y_k)f'(y_{in}) \]  
calculates its weight correction (used to update wjk),
\[ \Delta w_{jk} = \alpha \delta_k z_j \]  
calculating bias correction (used for updating w0k),
\[ \Delta w_{0k} = \alpha \delta_k \]
send δk to the layer unit below it.
Step 7. Each hidden unit (Zj, j = 1,...., p) adds up the input delta (from units on the top layer),
\[ \delta_{in} = \sum_{k} \delta_k w_{jk} \]  
multiplied by the derivative of the activation function for compute error information,
\[ \delta_j = \delta_{in}f'(z_{in}) \]  
calculating weight corrections (used to update vij),
\[ \Delta v_{ij} = \alpha \delta_j x_i \]  
and calculates the bias correction (used for update v0j),
\[ \Delta v_{0j} = \alpha \delta_j \]

Weights and bias updates
Step 8: Each unit of output (yk, k = 1, ...., m) updates the bias and weights (j = 0,...., P):
\[ w_{jk}^{\text{new}} = w_{jk}^{\text{old}} + \Delta w_{jk} \]  
\[ v_j^{\text{new}} = v_j^{\text{old}} + \Delta v_j \]

Each hidden unit \((Z_j, j = 1, \ldots, P)\) updates the weights and bias \((i = 0, \ldots, n)\):

Step 9: Stop condition test.

Application algorithm
Step 0: Initialize the weights \(V\) and \(W\)
Step 1: For each input vector perform steps 2-4
Step 2: For \(i = 1, \ldots, n\) activation set for the input unit \(x_i\).
Feedforward
Step 3: for each hidden unit \((z_j, j = 1, \ldots, p)\) add the weighting input signal,

\[ z_{-in} = v_{kj} + \sum_{i=1}^{x} x_{i} v_{ij} \]  

activation function, calculates the output signal:

\[ z_{j} = f(z_{-in}) \]

Step 5: Each output unit \((y_k, k = 1 \ldots \ldots m)\) adds up the input signals,

\[ y_{-in} = w_{0k} + \sum_{j} z_{j} w_{jk} \]  

activation function, calculates the output signal:

\[ y_{k} = f(y_{-in}) \]

The author designed the system in 3 components, namely the master file input, new image input and output. The following will be described by component. The input of the master file (Input) is received in the form of an image with the .bmp format, the image will be normalized into a grayscale image, then from the image the weights of \(v\) and \(w\) will be obtained by using Backpropagation training which can be seen in Figure 2. In Figure 3, the new image input will be processed using the application procedure in the Backpropagation training using the weights \(v\) and \(w\) in the master file image, and the savings are the result of the pattern calculation process from the input image master file and the new image process. Figure 4 shows the process of the image similarity percentage in the master file and the new image at the output system.

![Figure 2. Master Image Input Process](image1)

![Figure 3. New Image Input Process](image2)

![Figure 4. Image Similarity Detection Process](image3)

2.1 Flowchart Backpropagation
Backpropagation training includes 3 phases as shown in Figure 5. The first stage is Feedforward, the second stage is Backpropagation, and the third phase is weight modification.
Phase 1: The input signal is propagated to the hidden screen using the predefined activation function. The output of each hidden screen unit (zi) is then propagated forward to the hidden screen above it using the predefined activation function. As long as it has not produced output (yk), the process returns to the beginning.

Results Output (yk) is compared with the target to be achieved (tk). The result of the tk-yk difference is an error that occurs. If the error is less than the specified tolerance limit, the iteration is terminated. If the error is still greater than the tolerance limit, the weight of each line in the network will be modified to reduce the error that occurs. Figure 6 is a forward propagation flowchart.
Phase 2: Backpropagation
The validity of $t_k-y_k$ is calculated from the factor $\delta_k (k = 1,2,\ldots, m)$ and is used to transmit the error to all hidden units that are directly connected to the unit ($y_k$). The factor $\delta_k$ is used to change the weight of the line directly connected to the output unit. Changes in all line weights derived from hidden units in the lower screen are obtained by calculating the factor $\delta_j$ for each hidden layer. This stage is carried out continuously until all the factors $\delta$ in the hidden unit that are directly related to the input unit are accounted for.

Phase 3: Change in weight
The change in weight of all lines is used to factor $\delta$ of neurons in the layer above it. Factors are calculated to change the weight of all lines. The factor is calculated to modify the weights of all lines.

The three phases of Backpropagation training are repeated until the stopping condition reaches the target. In general, the termination conditions that are often used are seen from the number of iterations and errors.
3. Result and Discussion

3.1 Interface Design
On the main system form, there are 2 main menus and a page control. Page control is used to perform the image similarity detection process. The design of the image similarity detection form can be seen in Figure 7.

![Figure 7. Image Similarity Detection Form](image)

The following buttons and functions are found in the image similarity detection form:

- **Load**: used to select the master image with the bitmap format (.bmp) that will be used, after the user selects the image, the image will be displayed on the input image form.
- **Weight v and w**: used to calculate the weight v and w after the image is selected and also used as a storage of the weight results v and w that have been obtained.
- **New Image**: used to select a new image with the bitmap format (.bmp) that will be compared, after the user selects the image, the image will be displayed on the output image form.
- **Image Similarity**: used to run the image similarity detection process.
- **Exit**: used to exit the form.

3.2 Analysis
In this study, there are several things that will be analyzed, namely: Analysis of Pattern Recognition of grayscale images with 8-bit, 16-bit, and 24-bit .bmp formats.

| Size (pixel) | 3 | 8 | 128 | 256 | 512 | 1024 |
|-------------|---|---|-----|-----|-----|------|
| 3x2         | 100% | 100% | 100% | 100% | 100% | 90%  |
| 5x5         | 100% | 100% | 100% | 100% | 98.4% | 91.2% |
| 10x10       | 100% | 100% | 100% | 99.8% | 99.4% | 90.9% |
| 20x20       | 100% | 100% | 100% | 99.7% | 99.4% | 90.9% |
| 50x50       | 100% | 100% | 100% | 99.75% | 97.68% | 75.16% |
| 100x100     | 100% | 100% | 100% | 99.73% | 88.42% | 73.77% |

Table 1. Experiment Results for Image Pattern Recognition with 8-bit
From the results of system testing in Table 1, it can be seen clearly that the images tested are of various sizes and lots of z (hidden screen). The results show that not all sizes can be recognized 100%. At a size of 3x2 with lots of z = 3 to 512 the image patterns can be recognized 100%, at sizes 5x5, and 20x20 the image patterns can be recognized with lots of z to 256, and at pattern sizes of 10x10, 50x50, and 100x100 can only be recognized by many z up to 128. It can be concluded from the table above that the larger the image size and the more z in the image, the pattern is not 100% recognized.

The system test results in Table 2 can be seen clearly that the tested images are of various sizes and many z (hidden screen). The results show that not all sizes can be recognized 100%. At 3x2 and 5x5 sizes with lots of z = 3 to 512, the image pattern can be recognized 100%. The size of 10x10 image patterns can be recognized with lots of z = 3 to 256, and at a pattern size of 20x20 can only be recognized with lots of z up to 128. In testing, there is a difference, namely in testing images measuring 50x50 and 100x100. The image size is 50x50 the pattern can be recognized 100% with lots of z = 8 to 128, while in images that are 100x100 only at lots of z = 8.

Table 2. Experiment Result with 16-bit Image Pattern Recognition

| Size (pixel) | Many z (hidden screen) |
|--------------|-------------------------|
|              | 3          | 8          | 128         | 256         | 512         | 1024        |
| 3x2          | 100%       | 100%       | 100%        | 100%        | 97.2%       | 99.9%       |
| 5x5          | 100%       | 100%       | 100%        | 100%        | 97.2%       | 99.9%       |
| 10x10        | 100%       | 100%       | 100%        | 100%        | 99.3%       | 99.3%       |
| 20x20        | 100%       | 100%       | 100%        | 99.6%       | 88.77%      | 86.22%      |
| 50x50        | 85.64%     | 100%       | 99.8%       | 85.9%       | 71.87%      | 70.15%      |
| 100x100      | 85.14%     | 99.99%     | 99.67%      | 90.45%      | 70.15%      | 70.15%      |

Table 3 shows the results of system testing using various sizes (pixel) and hidden screens (z). The results show that not all sizes can be recognized 100%. At sizes 3x2 and 5x5 with lots of z = 3 to 512, the image pattern can be recognized 100%, at 10x10 the image pattern can be recognized with lots of z = 3 to 8, and at a pattern size of 20x20 can only be recognized with lots of z up to 128. Patterns measuring 50x50 can only be recognized by lots of z = 8 to 128, whereas patterns at 100x100 can only be recognized by using lots of z = 3.

Table 3. Experiment Results for 24-bit Image Pattern Recognition

| Size (pixel) | Many z (hidden screen) |
|--------------|-------------------------|
|              | 3          | 8          | 128         | 256         | 512         | 1024        |
| 3x2          | 100%       | 100%       | 100%        | 100%        | 90.28%      | 71.14%      |
| 5x5          | 100%       | 100%       | 100%        | 100%        | 90.28%      | 71.14%      |
| 10x10        | 100%       | 99.8%      | 99.6%       | 99.4%       | 99.3%       | 99.3%       |
| 20x20        | 100%       | 100%       | 99.9%       | 92.65%      | 86.62%      | 86.62%      |
| 50x50        | 85.75%     | 100%       | 99.81%      | 86.9%       | 72.83%      | 72.83%      |
| 100x100      | 100%       | 85.22%     | 99.65%      | 99.73%      | 90.28%      | 71.14%      |

4. Conclusions
Artificial neural network with Multilayer Perceptron architecture can be used as a grayscale image pattern recognition seen from the analysis results. From the experiments that the author has done, it can be stated that not all two image patterns are the same 100% similar to the human eye which sees the same two image patterns. The inequality of the two image patterns is the same because of the many hidden screens (z) on the Multilayer Perceptron. The more hidden screens (z) and the larger the image size being tested, the results are far from 100% or the pattern recognition is not perfect, the tolerance limit for many screens in pattern recognition is at the 256 limit. The inaccuracy of image pattern recognition is also due to the large number of colors in the two images, and the difference in
the number of colors between the master image and the new image. The difference in the size of the master image and the new image also affects the inaccuracy of pattern recognition. In this research, the image pattern recognition that is tested is only up to the image that has a size (pixel) of 100x100. To investigate further, image pattern recognition is not only limited to 100x100 but can be developed with a system that can quickly test large images such as 1024x768 and not only grayscale images but with color images.

References

[1] Aljarah I, Faris H, Mirjalili S, Al-Madi N (2018) “Training radial basis function networks using biogeography-based optimizer”. Neural Comput Appl 29(7):529–553. Fausett, Laurene(1994),
[2] Chen J-F, Do QH, Hsieh H-N (2015). “Training artificial neural networks by a hybrid PSO-CS algorithm”. Algorithms 8(2):292–308.
[3] Dhaneswara, Giri, Veronika s. Moertini. (2004). Jaringan Saraf Tiruan Propagasi Balik Untuk Klasifikasi Data . Jurusan Ilmu Komputer, Universitas Katolik Parahyangan, Bandung.
[4] Eny Maria et al., “Segmentasi Citra Digital Bentuk Daun Pada Tanaman Di Politani Samarinda Menggunakan Metode Thresholding”, JURTI, Vol.2 No.1, Juni 2018, ISSN: 2579-8790.
[5] Fausett, L. (1994). Fundamental of Neural Network : Architecture, Algorithms,and Applications. New Jersey : Prentice Hall, Inc.
[6] Hermawan, Arief. (2006). Jaringan Saraf Tiruan Teori & Aplikasi. Yogyakarta: Andi.
[7] Jose M. Chaves-González, Miguel A., Juan M.Sánchez-Pérez.(2010) "Detecting skin in face recognition systems: A colour spaces study". Digital Signal Processing 20: p 806–823.
[8] Jong Jek Siang, (2004). “Jaringan Saraf Tiruan & Pemrogramanya Menggunakan MATLAB”, Penerbit Andi, Yogyakarta.
[9] Nugroho, Setyo (2004). Penerapan Algoritma Quickprop pada Jaringan Syaraf Tiruan untuk Mendeteksi Wajah Manusia.Jurusan Teknik Informatika, STMIK STIKOM Balikpapan.
[10] Puspitanigrum, Diyah. (2006). Pengantar Jaringan Saraf Tiruan. Yogyakarta: Andi.
[11] Purnomo Mauridhi Hery, Muntasa Arif, “ Konsep Pengolahan Citra Digital dan Ekstraksi Fitur “, Graha Ilmu Yogyakarta, 2010.
[12] Rinaldi Munir, (2004) “Pengolahan Citra Digital dengan Pendekatan Algoritrik”, Informatika Bandung.
[13] Restu Poetra Alqurni et al, (2016). “Pengenalan Tanda Tangan Menggunakan Metode Jaringan Saraf Tiruan Perceptron Dan Backpropagation”, Techno.COM, Vol. 15, No. 4, November: 352-363.