Performance Analysis of Artificial Neural Networks used for Color Image Recognition and Retrieving

Jamil Al-Azzeh¹, Ziad Alqadi², Mohammed Abuzalata³

¹Computer Engineering Department, Al Balqa’a Applied University, Amman, 11134, Jordan
E-mail: azzehjamil@gmail.com

²Computer Engineering Department, Al Balqa’a Applied University, Amman, 11134, Jordan
E-mail: Natalia_maw@yahoo.com

³Computer Engineering Department, Al Balqa’a Applied University, Amman, 11134, Jordan
E-mail: abuzalata@bau.edu.jo

Abstract: Image recognition refers to methods and technologies that identify digital images such as places, logos, people, objects, buildings, and several other variables in images. Users are sharing vast amounts of data through applications, social networks, and websites. The large volume of digital data is being used by companies to deliver better and smarter services to the people accessing it. In this paper we will introduce a methodology of image recognition and retrieval process based on using ANN as a recognizer capable to identify digital image index. ANN will be implemented and tested in order to select an optimal ANN which must give the accurate image index with minimum MSE and maximum PSNR.

Keywords: Features, ANN, activation function, training function, MSE, PSNR.
1- Introduction

1-1 Artificial Neural Network

Artificial neural networks (ANN) [1-35] is a set of fully connected neurons as shown in figure (1), each neuron has an input and a set of weights, the output is generated by applying the summation process and according to the selected activation function the output is generated.

![ANN Sample Example](image)

**Figure (1): ANN sample example**

ANN is an important mathematical tool, and it is applicable for various industrial and engineering applications, such as color image recognition and retrieval [2], [3].

To maximize the efficiency of the used ANN (Getting the calculated output with zero errors between the target and the calculated output) and minimum processing time) we have to consider the following factors [4], [5]:

- Defining the input data set.
- What are the calculated targets to be achieved?
- Selecting ANN architecture (How many layers, and how many neurons in each layer).
- Selecting the training function.
- For each layer what is the activation function to be used? (Logsig, tansig, or linear).
- What is the acceptable error value? (To be closed to zero).
- How many training cycles to be used? (one training cycle includes a forward phase to calculate the neuron outputs, checking the error, if the error is not acceptable process the back word process to adjust the neurons weights).

- Select a starting values for the weights (easily is to initialize all the weights to zero).

Taking the above factors into consideration, using ANN as a recognition or retrieval tool we have to apply the following tasks:

- ANN creation
  1) Setting the input data set.
  2) Setting the targets.
  3) Creating ANN using a selected architecture.
  4) Setting all the weights to zero.
  5) Selecting some ANN parameters (Goal (error) =0, Number of training cycles to be used (epochs)).
  6) Raining ANN.
  7) If the error equal zero, save ANN to be used for recognition, otherwise adjust the number of cycles, or adjust ANN architecture and repeat this phase.

- ANN processing
  This phase is being used for image recognition or retraining and it can be implemented applying the following steps:
   1) Load ANN.
   2) Select color image features to be used as an input.
   3) Get the image index by running ANN.
   4) Use the index to recognize or retrieve the image from the images library or data base.

1-2 Color image features.

Digital color image is a 3D matrix as shown in figure (2), the first matrix represents the red color, and the second one represents the green color, while the third one represents the blue color[5], [6].
Color image usually has a huge size, which makes it difficult to retrieve or recognize the image pixel by pixel, for example the image shown in figure (3) has a size of $384 \times 512 \times 3 = 589824$ pixel, this image requires $0.075000$ seconds to be recognized and this time is high and must be reduced in order to achieve efficient method of recognition.

To enhance the process of image recognition or retrieval we can use a method to extract small number of values, which form an image vector of features, then this vector can be passed to ANN to get the image index, which can be easily used to recognize or retrieve the image as shown in figure (4).
Many methods are now used based on local binary pattern (LBP) such as center symmetric LBP (CSLBP) [7], [8], reduced LBP [9]. Figure (5) illustrates how LBP operator for a pixel can be calculated:

In our experimental part we will use the following method of image features extraction [5], [10], and [11], which is shown in figure (6):

![LBP Calculation](image)

**Figure (5): LBP calculation**

![LBP Operator](image)

**Figure (6): The used method to calculate LBP operator**

### 2- Implementation and Experimental Results

#### 2-1 Color Image Features

The above mentioned method of color image features was implemented using various color images in types and sizes; table (1) shows a sample of these features:
Table (1): Color images features

<table>
<thead>
<tr>
<th>Image number</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Image number</td>
</tr>
<tr>
<td>1</td>
<td>139375</td>
</tr>
<tr>
<td>2</td>
<td>114516</td>
</tr>
<tr>
<td>3</td>
<td>3640245</td>
</tr>
<tr>
<td>4</td>
<td>88123</td>
</tr>
<tr>
<td>5</td>
<td>132757</td>
</tr>
<tr>
<td>6</td>
<td>115642</td>
</tr>
<tr>
<td>7</td>
<td>116967</td>
</tr>
<tr>
<td>8</td>
<td>127722</td>
</tr>
<tr>
<td>9</td>
<td>129888</td>
</tr>
<tr>
<td>10</td>
<td>53737</td>
</tr>
</tbody>
</table>

From table (1) we can see that the array vector for each image is a unique, thus it can be used as a key or signature to recognize or retrieve the image from images database.

The features in table (1) must be reorganized to form the input data set to ANN, by creating a 4 by n matrix (n number of images in the data base), then the data set must by divided by a big number in order to avoid receiving all the time the output 1 when applying tansig or logsig activation functions as shown in figures (7) and (8):

Columns 1 through 6

<table>
<thead>
<tr>
<th>139375</th>
<th>114516</th>
<th>3640245</th>
<th>88123</th>
<th>132757</th>
<th>115642</th>
</tr>
</thead>
<tbody>
<tr>
<td>3879</td>
<td>18986</td>
<td>61727</td>
<td>27788</td>
<td>5034</td>
<td>12729</td>
</tr>
<tr>
<td>523</td>
<td>623</td>
<td>553</td>
<td>2116</td>
<td>863</td>
<td>1751</td>
</tr>
<tr>
<td>4176</td>
<td>13371</td>
<td>13107</td>
<td>29973</td>
<td>9151</td>
<td>17573</td>
</tr>
</tbody>
</table>

Columns 7 through 10

<table>
<thead>
<tr>
<th>116967</th>
<th>127722</th>
<th>129888</th>
<th>53737</th>
</tr>
</thead>
<tbody>
<tr>
<td>9555</td>
<td>4134</td>
<td>9710</td>
<td>6410</td>
</tr>
<tr>
<td>2276</td>
<td>1483</td>
<td>251</td>
<td>3165</td>
</tr>
<tr>
<td>19724</td>
<td>14661</td>
<td>7541</td>
<td>84654</td>
</tr>
</tbody>
</table>

Figure (7): Arranging features matrix
Figure (8): Dividing features matrix by 1000000

2-2 ANN Implementation

In this part we will introduce procedures needed to create an optimal ANN used to recognize or retrieve color image using its features. Here optimality means creating ANN with the following features:

- **Accuracy**: Moving mean square error (MSE) between the target output (T) and the calculated output (image index) (CO) to zero or maximizing the value of peak signal to noise ratio (PSNR), these parameters can be calculated as shown in following equations:

\[
MSE = \frac{\sum_{i=1}^{n} ABS(T(i) - CO(i))^2}{n}
\]

\[
PSNR = 10 \times log10\left(\frac{10^2}{MSE}\right)
\]

\[
n = \text{number of outputs}
\]

- **Minimum training time**

To achieve the above mentioned parameters we have to be carefully when answering the following questions:

- How many layers in the required ANN?
- How many neurons in each layer?
- Which activation function to select for each layer? (for the output layer it must be linear and for other layers it can be logsig or tansig).
- Which training function to be used?

We have to select one of the following training functions:

1) **Trainlm**: Levenberg-Marquardt back propagation training function.
2) **Traingd**: Gradient descent back propagation training function.
3) **Traingdm**: Gradient descent with momentum back propagation training function.
4) **Trainingdx**: Gradient descent with momentum and adaptive learning rate back propagation training function.

- How many training cycles to be used for ANN training?
- What are the neurons weights? (Initially it is better to initialize them to zeros).
- What is the value of training rate? (Usually used to adjust the weights during the training process).

To answer the above questions we used the following matlab code several times varying ANN parameters:

```matlab
clear all, clc

%Load input data
load a
idata=a/1000000; target=1:10;
tic

%Create ANN
netz=newff(minmax(idata),[4 1],{'tansig','purelin'},'trainlm');

%Initialize ANN parameters
netz=init(netz); netz.trainParam.goal=0; netz.trainParam.epochs=20000;
netz.trainParam.lr=0.002;

%Train ANN
netz=train(netz,idata,target)
toc

%Run ANN
y=(sim(netz,idata))
MSE=0;
for i=1:10
    MSE=MSE+((abs(target(1,i)-y(1,i))).^2);
end
MSE=MSE./10
PSNR=10*log10((10.^2)/MSE)
```

**Experiment 1:**

Fixing training rate to 0.2 and training cycles to 1000

Table (2) shows the results of the first experiment (Number of layer 2[input with 4 neurons and output layer with one neuron], training rate=0.2 maximum training cycles=1000 varying the training rate:
Table (2): Experiment 1 results

<table>
<thead>
<tr>
<th>Training function</th>
<th>Activation function</th>
<th>Training cycles</th>
<th>MSE</th>
<th>PSNR</th>
<th>Training time(seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>trainlm</td>
<td>tansig</td>
<td>42</td>
<td>8.3357e-027</td>
<td>280.7906</td>
<td>0.615000 (optimal)</td>
</tr>
<tr>
<td>trainlm</td>
<td>logsig</td>
<td>89</td>
<td>2.7067e-025</td>
<td>265.6757</td>
<td>0.771000</td>
</tr>
<tr>
<td>traingd</td>
<td>tansig</td>
<td>1000</td>
<td>4.8297</td>
<td>13.1608</td>
<td>2.673000</td>
</tr>
<tr>
<td>traingd</td>
<td>logsig</td>
<td>1000</td>
<td>4.4665</td>
<td>13.5003</td>
<td>2.353000</td>
</tr>
<tr>
<td>traingdm</td>
<td>tansig</td>
<td>1000</td>
<td>4.5299</td>
<td>13.4391</td>
<td>2.489000</td>
</tr>
<tr>
<td>traingdm</td>
<td>logsig</td>
<td>1000</td>
<td>4.6187</td>
<td>13.3548</td>
<td>2.419000</td>
</tr>
<tr>
<td>traingdx</td>
<td>tansig</td>
<td>1000</td>
<td>2.8331</td>
<td>15.4773</td>
<td>2.346000</td>
</tr>
<tr>
<td>traingdx</td>
<td>logsig</td>
<td>1000</td>
<td>3.8044</td>
<td>14.1971</td>
<td>2.422000</td>
</tr>
</tbody>
</table>

From table (2) we can see that the optimal ANN is with 2 layers and the activation function to be used for input layer is tansig, while the training function must be trainlm. Figure (9) shows MSE using tansig, while figure (10) shows MSE using logsig.

Figure (9) : MSE using tansig
Experiment 2:

Fixing training rate to 0.2 and training cycles to 20000

Table (3) shows the results of this experiment

Number of layer 2[input with 4 neurons and output layer with one neuron] training rate=0.002 maximum training cycles=20000

Table (3): Experiment 2 results

<table>
<thead>
<tr>
<th>Training function</th>
<th>Activation function</th>
<th>Training cycles</th>
<th>MSE</th>
<th>PSNR</th>
<th>Training time(seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>trainlm</td>
<td>tansig</td>
<td>120</td>
<td>4.3293e-028</td>
<td>280.7906</td>
<td>0.809000</td>
</tr>
<tr>
<td>trainlm</td>
<td>logsig</td>
<td>235</td>
<td>2.7238e-026</td>
<td>293.6359</td>
<td>1.199000 (optimal)</td>
</tr>
<tr>
<td>trainlm</td>
<td>logsig</td>
<td>20000</td>
<td>4.4637</td>
<td>13.5030</td>
<td>42.229000</td>
</tr>
<tr>
<td>trainlm</td>
<td>tansig</td>
<td>20000</td>
<td>4.4849</td>
<td>13.4824</td>
<td>43.083000</td>
</tr>
<tr>
<td>trainlm</td>
<td>logsig</td>
<td>20000</td>
<td>4.8532</td>
<td>13.1397</td>
<td>42.978000</td>
</tr>
<tr>
<td>trainlm</td>
<td>tansig</td>
<td>20000</td>
<td>1.0136</td>
<td>19.9414</td>
<td>47.332000</td>
</tr>
<tr>
<td>trainlm</td>
<td>logsig</td>
<td>20000</td>
<td>1.3099</td>
<td>18.8275</td>
<td>47.544000</td>
</tr>
</tbody>
</table>

Here we can see that logsig gives a optimal ANN
Experiment 3

Here we will use only trainlm training function, because it is the best to be used, and we vary the number of layers:

Table (4) shows the results of this experiment (training rate =0.2)

From table (4) we can see that adding a new hidden layer to ANN architecture will improve ANN accuracy by increasing PSNR and decreasing MSE but with an extra time for training and extra space to save ANN.

Table (4): Experiment 3 results

<table>
<thead>
<tr>
<th>Layers</th>
<th>Activation function</th>
<th>Training time(seconds)</th>
<th>ANN size(bytes)</th>
<th>MSE</th>
<th>PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 1</td>
<td>L, LN</td>
<td>0.684000</td>
<td>26785</td>
<td>1.5399e-028</td>
<td>298.1252</td>
</tr>
<tr>
<td>4 1</td>
<td>T, LN</td>
<td>0.762000</td>
<td>26785</td>
<td>5.1087e-028</td>
<td>292.9169</td>
</tr>
<tr>
<td>4 4 1</td>
<td>L, L, LN</td>
<td>0.798000</td>
<td>33487</td>
<td>3.0545e-028</td>
<td>295.1506</td>
</tr>
<tr>
<td>4 4 1</td>
<td>L, T, LN</td>
<td>0.774000</td>
<td>33487</td>
<td>3.4699e-027</td>
<td>284.5969</td>
</tr>
<tr>
<td>4 4 1</td>
<td>T, L, LN</td>
<td><strong>0.611000</strong></td>
<td><strong>33487</strong></td>
<td><strong>7.5021e-029</strong></td>
<td><strong>301.2482</strong> (optimal)</td>
</tr>
<tr>
<td>4 4 1</td>
<td>T, T, LN</td>
<td>0.715000</td>
<td>33487</td>
<td>1.0657e-023</td>
<td>249.7238</td>
</tr>
<tr>
<td>4 8 1</td>
<td>L, L, LN</td>
<td>0.643000</td>
<td>34095</td>
<td>8.4985e-023</td>
<td>240.7066</td>
</tr>
<tr>
<td>4 8 1</td>
<td>L, T, LN</td>
<td>0.849000</td>
<td>34095</td>
<td>8.5217e-029</td>
<td>300.6948</td>
</tr>
<tr>
<td>4 8 1</td>
<td>T, L, LN</td>
<td>0.648000</td>
<td>34095</td>
<td>1.7247e-022</td>
<td>237.6328</td>
</tr>
<tr>
<td>4 8 1</td>
<td>T, T, LN</td>
<td>0.777000</td>
<td>34095</td>
<td>2.0593e-025</td>
<td>266.8627</td>
</tr>
</tbody>
</table>

Conclusion

Color image features were extracted and passed to ANN. It was shown that selecting the following will lead to achieving optimal ANN by mean of maximizing:

- Using the training function trainlm improve ANN performance.
- It is preferable to use input layer with number of neurons equal to the number of inputs in the input data set.
- For hidden layers we have to use logsig or tansig activation function.

- Adding a hidden layer will improve ANN performance.

References


27. Dmitriy Skopin and Jamil Al-Azzeh; Automated Demodulation of Amplitude Modulated Multichannel Signals with Unknown Parameters Using 3D Spectrum Representation Research Journal of Applied Sciences, Engineering and Technology, Maxwell Scientific Publication June 05, 2016; (SJR indicator = 0.15).