# Chapter V
# Damageless Watermark Extraction Using Nonlinear Feature Extraction Scheme Trained on Frequency Domain

**Kensuke Naoe**
*Keio Univesity, Japan*

**Yoshiyasu Takefuji**
*Keio University, Japan*

## ABSTRACT

*In this chapter, we propose a new information hiding and extracting method without embedding any information into the target content by using a nonlinear feature extraction scheme trained on frequency domain. The proposed method can detect hidden bit patterns from the content by processing the coefficients of the selected feature subblocks to the trained neural network. The coefficients are taken from the frequency domain of the decomposed target content by frequency transform. The bit patterns are retrieved from the network only with the proper extraction keys provided. The extraction keys, in the proposed method, are the coordinates of the selected feature subblocks and the neural network weights generated by the supervised learning of the neural network. The supervised learning uses the coefficients of the selected feature subblocks as the set of input values, and the hidden bit patterns are used as the teacher signal values of the neural network, which is the watermark signal in the proposed method. With our proposed method, we are able to introduce a watermark scheme with no damage to the target content.*

## INTRODUCTION

In this chapter, we present a new model of digital watermark that does not embed any data into the content, but is able to extract meaningful data from the content. This is done by processing the coefficients of the selected feature subblocks to the trained neural network. This model trains a neural

network to assign predefined secret data, and use the neural network weight and the coordinates of the selected feature subblocks as a key to extract the predefined code. This means that it would not damage the content at all. The proposed method is an improvement from the paper of our research project, which was published before (Ando & Takefuji, 2003).

In Section 2, we discuss the background surrounding digital watermarks, frequency transformation, and neural networks. We demonstrate the characteristics, and discuss what techniques are useful for implementing digital watermarks. In Section 3, we propose the method of damageless watermark embedding and extraction for still image. In Section 4, we provide experiment results for testing its robustness and fragileness. In Section 5 and 6, we conclude with a discussion of the proposed method, and indicate some future works of the proposed method.

## BACKGROUND

In this section, we discuss the background surrounding digital watermarks, and we go deeply to the backgrounds and researches in frequency transformation and neural networks, which consist of important modules for the proposed method.

### General Background Surrounding Digital Watermarks

Recently, with the rapid progress of information technologies and the emergence of the Internet, digital multimedia contents are easily distributed on the network. This circumstance helped to open digital contents to the public without difficulty, even for ordinary computer users, but also helped illegal distribution and copying of contents. Due to the characteristics of digital contents, digital contents are easy to make an exact copy and to alter the content itself. This became a main concern for authors, publishers, and legitimate owners of the contents. Therefore, digital watermark became a key technology for protecting the copyrights. Digital watermark protects unauthorized change of the contents and assures legal uses for its copyright.

There are several ways to protect digital content. One example is to encrypt the content and to share the decryption key between the author and the observer. But this method prevents other observers without a decryption key from accessing the content. This feature avoids a free distribution and circulation of the content through the network, which, most of the time, is not desirable to the author of the content. Digital watermark only embeds data to the content, and this feature does not avoid the distribution of the content.

Watermarking techniques are one technique of information hiding techniques (Katzenbeisser & Petitcolas, 2000). The research in information hiding has a history (Kahn, 1996), namely, the researches in digital watermark and steganography have been active. Both are very similar, but the applications are different (Reither & Rubin, 1998). Digital watermark can be classified in several ways. The first classification is by the perceptibility of the watermark signal to humans. A perceptible watermark has various usages but because it limits the utility of the content, most of the research in this area has focused on imperceptible watermarking techniques. Digital watermark is often embedded imperceptibly to human receptors to avoid contaminating the content. For imperceptible images, the human visual system (HVS) model is often used. There are many still image watermark researches that make use of HVS model (Delaigle, De Vleeschouwer, & Macq, 1998; Kim, Byeong, & Choi, 2002; Reither & Rubin, 1998; Swanson, Zhu, & Tewfil, 1996; Westen, Lagendijk, & Biemond., 1996). For imperceptible audio, a psychoacoustic model is often used. The basic idea of the psychoacoustic model is that human ears are insensitive to the change in phase and amplitude, but very sensitive to the change in the time domain. Also, humans have

a limitation to high frequency domain and low frequency domain (Cox, Miller, & Muttoo, 2002). There are many researches for audio watermark and many of them use a psychoacoustic model for implementations (Baumgarte, Ferekidis, & Fuchs, 1995; Boney, Tewfik, & Hamdy, 1996; Gruhl, Lu, & Bender, 1996; Wolfe & Godsill, 2000).

Most imperceptible still image watermark methods use the HVS model in order to embed data into frequency domains after the frequency transformation of multimedia content, such as DFT, DCT, and DWT. Perceptible watermarks are sometimes used, but it limits the use of the images. Therefore, main concern in this research area has focused on imperceptible watermarks. In general, digital watermark is a technique to conceal a code imperceptibly to the content to anybody who observes it, and it is also difficult to remove from the content to protect its intellectual property rights.

The second classification of watermark technique is based on whether the original content data is used during the watermark extraction process. A watermark method that needs original content to extract a watermark from embedded content is called nonblind detection watermark. A watermark method that does not need original content and only needs the embedded content to extract a watermark is called blind detection watermark. This characteristic raises the difference between the extraction method and its robustness.

The third classification of watermark is based on its robustness to the attacks. Robust watermark is normally strong against illegal alteration of the content and it is difficult to remove watermark data from the content. Even if the attacker was able to remove the watermark data, the damage to the content is very intense. Therefore, robust watermark is often used for copyright protection. A watermark method using spread spectrum is widely known for its robustness (Cox, Kilian, Leightont, & Shamoon 1997). Spread spectrum has been deployed in many researches to strengthen the robustness of the watermark. There are many researches of robust watermarking techniques (Bender, Gruhl, & Morimoto, 1995; O'Ruanaidh & Pun, 1998) and watermarks for other media such as motion picture (Echizen, Yoshiura, Arai, Himura, & Takeuchi, 1999) and text data (Brassil, Low, Maxemchuk, & O'Gorman, 1994; Brassil & O'Gorman, 1996).

To the contrary, a method called fragile watermark is a watermark method that is usually weak to alteration of the content. The watermark data will be broken or diminished when the content data is damaged. Therefore, fragile watermark is often used for an integrity check of the content. Another watermark, known as semifragile watermark, has a characteristic of both robust and fragile watermark.

Recently, a watermark method using neural network has been spreading widely. Commonly seen methods in this area are the ones that use either DCT or DWT for decomposition of the image, and a back propagation neural network is used for embedding the watermark (Davis & Najarian, 2001; Hong, Shi, & Luo, 2004; Lou, Liu, J.L. & Hu, 2003; Zhang, Wang, N.C. & Xiong, 2002). Beside that, watermark methods using generalized Hebb (Pu, Liao, Zhou, & Zhang, 2004), RBF neural network (Liu & Jiang, 2005; Zhang, et al., 2003), full counterpropagation neural network (Chang & Su, 2005), Hopfield neural network (Zhang, & Zhang, 2004; Zhang, & Zhang, 2005), and many more are introduced. The use of neural network can be used to achieve both robust and fragile watermarking method.

But many of the methods that use neural network embed a watermark into the content data, which will alter the quality of data. Our method does not embed any data in the target content, which does not damage the content data at all. This characteristic is the biggest difference between the conventional methods.

## Background for Frequency Transformation

In the research area of information hiding, including digital watermark, orthogonal transformation is used for the watermark embedding process to embed a watermark signal into the frequency domain of the image. Embedding in the frequency domain is commonly used because this method is able to limit the damage to the content rather than embedding the signal to the image directly. Frequency analysis is a useful tool because with the analysis, it is able to understand which range of frequency band is the important frame of the image and which is not. According to human visual system (HVS), humans are sensitive to the changes in the low frequency domain and insensitive to the changes in the high frequency domain. Consequently, watermark signals are embedded in the higher frequency domain to achieve imperceptible watermarking. But embedding a watermark signal to the high frequency band has a risk of data loss in case of image compression that uses the feature of HVS systems, like JPEG compression.

There are several methods to transform an image to frequency domain, such as discrete Fourier transform (DFT), discrete cosine transform (DCT), and discrete wavelet transform (DWT). In our proposed method, DCT is used, so we will discuss it in more detail.

Discrete cosine transform is one method of orthogonal transformation. The image compression using DCT orthogonally transforms the small blocks in the image. Compression using discrete cosine transform is known to be very effective, and is employed in MPEG format and JPEG format.

Discrete cosine transform divides an image into *N***N* pixel small blocks. The transformation will be processed independently to the individual *N***N* blocks. When expressing the coefficients after the discrete cosine transform as $F(u,v)$, the discrete cosine transform function, and inverse discrete cosine transform function to the *N***N* block of two dimensional image signal $f(x,y)$ is as follows.

DCT function of *N***N* two dimensional image:

$$F(u,v)=\frac{2}{N}c(u)c(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}f(x,y)\cos\left\{\frac{(2x+1)u}{2N}\pi\right\}\cos\left\{\frac{(2y+1)v}{2N}\pi\right\}$$

$$\text{if } u=0 \text{ then } c(u)=\frac{1}{\sqrt{2}} \qquad \text{if } v=0 \text{ then } c(v)=\frac{1}{\sqrt{2}}$$

$$\text{if } u\neq 0 \text{ then } c(u)=1 \qquad \text{if } v\neq 0 \text{ then } c(v)=1$$

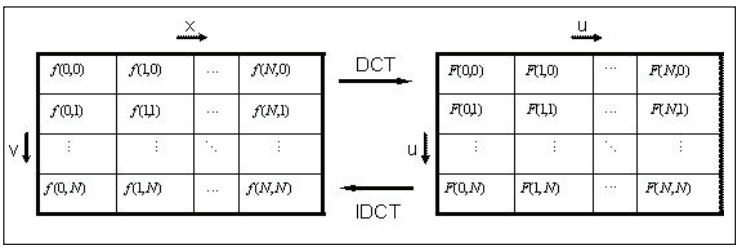
Inverse discrete cosine transformation function of *N***N* two dimensional image:

$$f(x,y)=\frac{2}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}c(u)c(v)F(u,v)\cos\left\{\frac{(2x+1)u}{2N}\pi\right\}\cos\left\{\frac{(2y+1)v}{2N}\pi\right\}$$

The relation between the image signal and the frequency coefficients are shown in Figure 1.

When the size of *N* is the bigger, the less inferior the image will be, but the calculation cost will increase proportional to *N*^2; ideal number of *N* is known to be 8. When the image is 512*512 pixels in size, the amount of 8*8 blocks will be 64*64, total of 4096 of 8*8 subblocks. Generalized discrete cosine transform function of *N* = 8 is:

*Figure 1. Relation of pixel value and DCT coefficients*



$$S(u,v)=\frac{1}{4}C(u)C(v)\sum_{m=0}^{7}\sum_{n=0}^{7}P(m,n)\cos\frac{(2m+1)u\pi}{16}\cos\frac{(2m+1)v\pi}{16}$$

$$C(u)C(v)\begin{cases}\text{if} & \frac{1}{\sqrt{2}} & \text{then} & u,v=0\\ \text{if} & 1 & \text{then} & u,v\neq 0\end{cases}$$

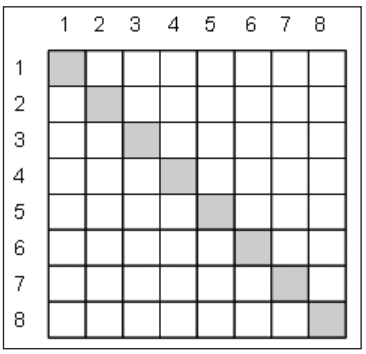
$$(u,v=0,1,2,\cdots,7)$$

As shown in Figure 1, the coefficient of $F(0,0)$ is the direct current and the other $F(u,v)$ is the alternate current. Coefficients closer to the direct current will be lower frequency value and closer to $F(N,N)$ will be higher frequency value. In general, low frequency values have more energy than high frequency values, and are known to be more affective to the composition of an image. In case of image compression, the region of high frequency band is omitted or quantized more strongly than the lower frequency band. Hence, embedding watermark signals to the high frequency band has a possibility of being omitted during the compression process, while embedding watermark signals to the lower frequency band will damage the frame of the original image. Considering the robustness of watermark, watermark signals are often embedded in the midfrequency band. The proposed method needs feature values from low to high frequency values, so values are taken diagonally from $F(0,0)$ to $F(N,N)$ as feature input values to the neural network.

## Background for Neural Network

A neuron is a nerve cell and the brain consists of many nerve cells. The neuron itself only has simple function of input and output of electric signal. But when these neurons organically connect to form a network, the network is capable of complex processing. Mathematical models of these networks are called neural networks, and processing these artificial neural networks on computers is called neural computing. Neural computing is a classic research topic and neural networks are known to be capable of solving various kinds of problems by changing the characteristics of neuron, synaptic linking, and formation of the network.

*Figure 2. DCT coefficients chosen diagonally in proposed method*




Binary model (Hopfield & Tank, 1985), sigmoid model (McCulloch & Pitts, 1943), radial basis function model (Moody & Darken, 1989; Poggio & Girosi, 1990), and competitive neuron model (Kohonen, 1982; Kohonen, 1995; Takefuji, Lee, & Aiso, 1992) are some of the models classified by the model of neuron units itself. A neural network can also be classified by its network formation. Recurrent model is a model in which neurons are connected mutually. Feed forward model is a model in which neurons are connected in a layer and signals are transmitted simply from input layer to output layer, whereas feedback model is a model similar to feed forward model, but it has a backward or recursive synaptic link. Both feed forward and feedback model sometime has a bias synaptic link.

The proposed method uses a multilayered perceptron model as a network model which has a feedforward synaptic link. Multilayered perceptron model was introduced by Rosenblatt (Rosenblatt, 1985). Two layer perceptron can classify data linearly but cannot classify data nonlinearly. Multilayered perceptron with more than three layers are known to have an approximation ability of a nonlinear function if properly trained, but then there was no ideal learning method for this kind of training. This model became popular when a training method called back propagation

learning, which uses generalized delta rule, was introduced by Rummelhart (Rummelhart, McClelland, & the PDP Research Group, 1986). Since then, this neural network model is often used to solve nonlinear classification problems. Other popular neural network models are RBF network model (Broomhead & Lowe, 1988), pulsed neural network model (Eckhorn, Reitboeck, Arndt, & Dicke, 1990; Johnson, 1994), second order neural network model (Maxwell, Giles, Lee, & Chen., 1986), chaotic neural network model (Aihara, 1990), Elman network model (Elman, 1994), and many others.

## Background for Multilayered Perceptron Model

Multilayered perceptron basically has a synaptic link structure with neurons between the layers but no synaptic link among the layer itself. Signal given to the input layer will propagate forwardly according to the synaptic weight of the neurons connected, and reaches to the output layer as shown in Figure 3.

For each neuron, input values are accumulated from a former layer, and outputs a signal value according to a certain function; normally sigmoid function is used. Sigmoid function has a graph like Figure 4, and this function is expressed as:

$$y = \frac{1}{1+\exp(-x)}$$

Each network connection has a network weight. The network weight from unit $i$ to unit $j$ is expressed as $W_{ij}$. The output value from the former layer is multiplied with this value. These output values of the former layer are summed and connected to the upper layer. Therefore, the output values for the output layer are determined by the network weight of the neural network, as shown in Figure 5. Consequently, to change the output value to a desired value, adjustment of these network weights is needed. Learning of the network is a process of conditioning the network weight with the corresponding teacher values.

## Background for Back Propagation Learning

Back propagation learning is the process of adjusting the network weights to output a value close to the values of the teacher signal values.

*Figure 3. Example of multilayered perceptron model*

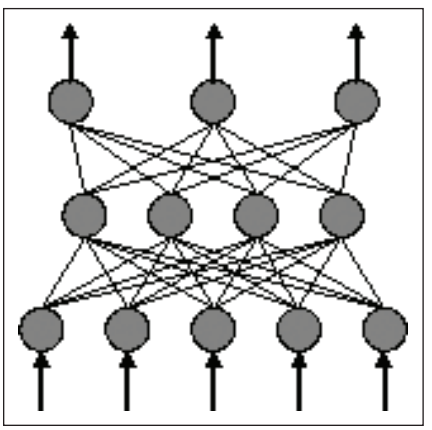


*Figure 4. Graph of sigmoid function*

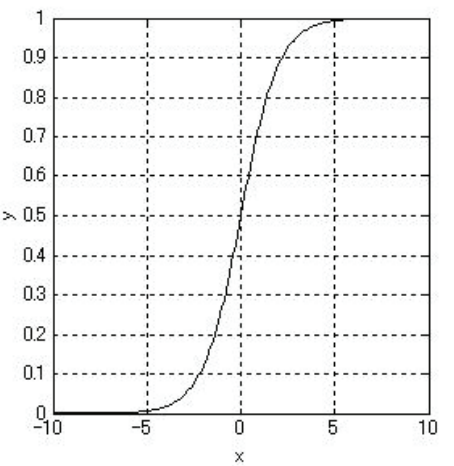
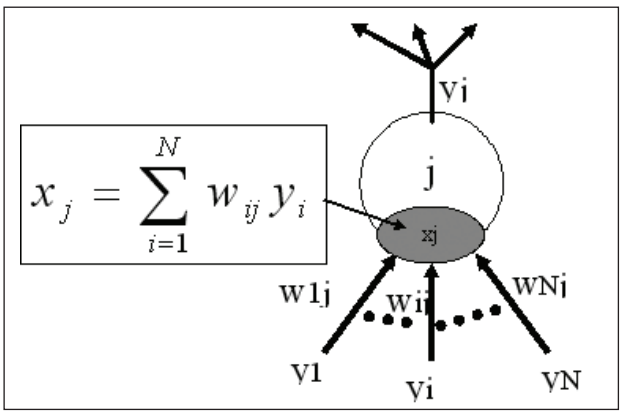


*Figure 5. Input values and network weights determinate the output value*




Back propagation learning is a supervised learning. In back propagation learning, teacher signal values are presented to the network. This training tries to lower the difference between the teacher signal values and the output values by changing the network weight. This learning is called back propagation because the changes of the network weight according to the difference in the upper layer propagate backward to the lower layer. This difference between the teacher signal values is called as error and often expressed as delta (δ).

When teacher signal $t_j$ is given to the unit $j$ of output layer, the error $\delta_j$ will be the product of the differential for sigmoid function of output unit $y_j$ and difference between the teacher signal $t_j$ and output value of unit $y_j$. So the function for calculating the error value for the output unit is:

$$\delta_j = (t_j - y_j) f'(y_j) = (t_j - y_j)(1 - y_j) y_j$$

On the other hand, calculation of error value for the hidden neurons is more complicated. This is because the error value of the output unit affects the error value for the hidden neuron. The error value for the hidden unit varies with the network weight between the hidden layer and output layer. In order to calculate the error value $\delta_i$ for hidden unit, error value $\delta_j$ of the output unit is used. So the function to calculate the error value $\delta_i$ for hidden unit $i$ is:

$$\delta_i = f'(y_i) \sum_{j=1} W_{ij} \delta_j = (1 - y_i) y_i \sum_{j=1} W_{ij} \delta_j$$

After calculating the error values for all units in all layers, then network can change its network weight. The network weight is changed by using the function:

$$\Delta W_{ij} = \eta \delta_j y_i$$

η in this function is called learning rate. Learning rate normally has a value between 0 and 1, and generally represents the speed of learning process. The lower the learning rate is the more gradual the learning process will be, and the bigger the learning rate is the more acute the learning process will be. Sometimes this parameter must be tuned for stable learning.

## Proposed Method

In this section, we present the hidden bit patterns embedding method to the neural network using the selected feature values from the image, and extraction method using the network weight

from the trained network with the selected feature values. With this algorithm, we are able to implement a function of digital watermark with no embedding of data into the content data. No embedding into the content implies no damage to the content data.

To accomplish our goals, we use values of a certain area in the frequency domain as input data to the neural network, and teach the network with a classification set of data. The classification data set will be the identification information that acts as watermark data. A trained neural network will have a network weight, and is used as the extraction key for the output data that is trained to approximate the watermark signal. With this model, the attacker cannot reproduce a watermark signal if he has only one of the input data set or network weights. The extractor must have both proper input data and network weights to induce a watermark signal from neural network.

With the proposed embedding method, we must decide the structure of neural network. The amount of neurons for input layer is decided by the number of pixels selected as feature values from the content data. The proposed method uses the diagonal values of the frequency domain from the selected feature subblock of the content data. For example, if the image is 512*512 pixels in size and the frequency transform is done by 8*8 DCT, then there will be 4096 subblocks being produced, and the neurons in the input layer will be 8. We select a unique feature subblock in obedience to the teacher signal. Diagonal 8 pixels of the selected subblock will be taken as input values for the neural network. In our proposed method, one bias neuron is added for better approximation also. Therefore, we will have nine neurons in the input layer for proposed method.

The output layer will be trained to output 1 or 0 as an output value. The amount of neurons for output layer is decided by the number of identification values or patterns to be embedded into the network. In the case where 32 unique identification values are to be embedded into the network, five sets of network with one output neuron must be prepared in order to embed five bits of data. This is because each network represents, for each digit of the binary watermark signal, respectively, and 5 binary digits are sufficient to represent 32 different values.

The adequate amount of neurons in the hidden layer necessary for an approximation, in general, is not known. So the number of neurons in the hidden layer will be taken at will. For better approximation, a bias neuron like in input layer can be introduced for the hidden layer too. After the neural network structure is decided, the process is moved on to the back propagation learning procedure.

This learning process uses watermark signal data as teacher signal, which is either 1 or 0, corresponding to the digit of the watermark. After the learning process, the network weights are converged to certain values. We use these network weights and the coordinates of selected feature subblocks as the extraction keys of the embedded watermark data.

For the extraction process, we will take the same neural network structure with the network weights generated in the embedding process. Only the proper input values of the selected feature subblocks will output the proper watermark signal. Proper input values are induced only when you know the proper coordinates of the subblocks for the corresponding watermark signal.

## Necessary Parameters for the Proposed Method

For embedding, there are two parameters to decide on. First is the number of class patterns, which are the identifier in the extraction process. The more the number of class patterns, the more data to be embedded, but a large number of class patterns will have a high calculation cost. Second parameter is

the coordinate of the subblock that you associate the patterns to. Coordinates determine the input values for the learning process, which generates a neural network weight that acts as an extraction key for watermark in the extraction process. These two parameters are stored as the extraction keys and handed to extractor of the watermark.

For extracting, two parameters are needed to extract a proper watermark signal. First is the neural network weights created in the embedding process. Second is the coordinates of the subblocks that were associated with the patterns. These network weights and coordinates of the subblocks are the extraction keys for the watermark signal. These parameters are handed as an extraction key from the proper user who embedded the watermark. Only with the presence of the proper network weights and the proper coordinates is one able to output the proper watermark signal.

We will discuss both embedding process and extraction process in detail in the following subsections.

## Embedding Process

Embedding steps consist of the following procedures:

1. Frequency transform of the image
2. Selection of the feature subblocks
3. Backpropagation learning process
4. Save the coordinates of the selected feature subblocks and converged network weights

### Frequency Transform of the Image (Process 1)

This procedure simply performs a frequency transformation of the image. If DCT is used for frequency transformation method, it transforms value in each pixel into DCT coefficient values.

### Selection of the Feature Subblocks (Process 2)

If user desires to embed 32 different identification patterns, then the same amount of unique subblocks must be chosen from the image. The feature subblocks can be selected randomly, or voluntarily by the user. Besides, sufficient number of networks must be prepared, which will be the number of binary digits to satisfy the identification values. In this case, five networks are enough to represent 32 different identification values.

### Backpropagation Learning Process (Process 3)

A set of learning processes for each network is constructed by the training of input data set for corresponding teacher signal value, respectively. If identification patterns are 32, a set of learning processes for the network is training the network with 32 different sets of input values of the corresponding teacher signals. This learning process is repeated until the output value satisfies a certain learning threshold value. This threshold value can be set flexibly according to the usage of the watermark. This learning must be done for all five neural networks.

### Save the Coordinates of the Selected Subblocks and Converged Network Weights (Process 4)

After the learning process for all the networks have converged, the coordinates of the selected feature subblocks and the values of network weights are saved. Extractor will use this information to extract a watermark signal in the extraction process.

## Extraction Process

Extraction step consists of the following procedures:

1. Frequency transform of the image.
2. Load the coordinates of selected feature subblocks and the values of network weights.
3. Feed forward the network with the input values from the selected feature subblocks.

## Frequency Transform of the Image (Process 1)

This procedure is an equivalent process as the frequency transform of the image in the embedding process. We simply perform a frequency transformation of the image. If DCT is used for frequency transformation method, it transforms the value of each pixel into DCT coefficient values.

## Load the Coordinates of Selected Feature Subblocks and the Values of Network Weights (Process 2)

Before the process, extractor must receive the coordinate data from the embed user. Extractor will use the coordinates of selected feature subblock and network weights as the extraction keys to extract the watermark signal. Knowing the coordinates of the feature subblocks will lead user to the proper input values for the embedded watermark signal. By knowing the network weights value, we can induce the structure of the network, and only proper network weights are able to output the proper watermark signal.

## Feed Forward the Network with the Input Values from the Selected Feature Subblocks (Process 3)

After constructing the neural networks with proper network weights, we examine the output value of the network with the input values induced from the feature subblocks. Each network is to output either 1 or 0 with the aid of threshold value for output unit. If the network weights and the input values are properly set, the trained network must be able to output the corresponding watermark

*Figure 6. Image of Lena*

*Table 1. Coordinates of feature subblocks and corresponding teacher signal*

| x-axis | y-axis | coordinate | teacher signal |
|---|---|---|---|
| 58 | 64 | (58,64) | 00000 |
| 34 | 7 | (34,7) | 00001 |
| 43 | 30 | (43,30) | 00010 |
| 44 | 9 | (44,9) | 00011 |
| 61 | 46 | (61,46) | 00100 |
| 40 | 20 | (40,20) | 00101 |
| 30 | 5 | (30,5) | 00110 |
| 42 | 12 | (42,12) | 00111 |
| 63 | 13 | (63,13) | 01000 |
| 27 | 43 | (27,43) | 01001 |
| 63 | 25 | (63,25) | 01010 |
| 34 | 5 | (34,5) | 01011 |
| 35 | 44 | (35,44) | 01100 |
| 25 | 55 | (25,55) | 01101 |
| 55 | 7 | (55,7) | 01110 |
| 35 | 26 | (35,26) | 01111 |
| 53 | 25 | (53,25) | 10000 |
| 32 | 33 | (32,33) | 10001 |
| 8 | 58 | (8,58) | 10010 |
| 10 | 42 | (10,42) | 10011 |
| 51 | 14 | (51,14) | 10100 |
| 18 | 5 | (18,5) | 10101 |
| 36 | 52 | (36,52) | 10110 |
| 9 | 28 | (9,28) | 10111 |
| 28 | 58 | (28,58) | 11000 |
| 4 | 64 | (4,64) | 11001 |
| 51 | 26 | (51,26) | 11010 |
| 36 | 22 | (36,22) | 11011 |
| 50 | 21 | (50,21) | 11100 |
| 64 | 24 | (64,24) | 11101 |
| 6 | 40 | (6,40) | 11110 |
| 47 | 39 | (47,39) | 11111 |

signal. We will use an example to explain these processes in the next subsection.

## Embedding of Watermark Signal

In this example, we will use TIFF format Lena image as in Figure 6, which is 512*512 pixels in size, as the target content data. We will embed 32 different identification values as a watermark signal. Therefore, five different neural networks must be prepared because 32 can be defined with five digits of binary numbers.

First, frequency transformation of DCT will be processed to the Lena image. 8*8 pixels subblocks are produced for 512*512 pixels Lena image. There will be a total of 4,096 subblocks produced. In terms of subblocks, it will be 64*64 subblocks are produced. Feature subblocks are selected randomly with the teacher signals correspondingly, and those coordinates of subblocks are saved as Table 1.

For example, before performing DCT, the original pixel values in subblock (47, 39) are as follows.

When DCT is processed, the transformed DCT coefficients values of the subblock (47, 39) are as follows.

For all 32 subblocks, eight diagonal DCT coefficient values are taken as feature values and treated as input values to the network. These values are then trained to a corresponding watermark signal value, which is the teacher signal of the

*Table 2. Value of R in subblock (47, 39)*

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 223 | 215 | 221 | 223 | 220 | 213 | 217 | 216 |
| 2 | 218 | 219 | 221 | 222 | 218 | 217 | 217 | 217 |
| 3 | 224 | 223 | 221 | 220 | 216 | 216 | 219 | 215 |
| 4 | 221 | 224 | 220 | 217 | 218 | 214 | 215 | 214 |
| 5 | 220 | 222 | 221 | 217 | 217 | 213 | 214 | 212 |
| 6 | 221 | 222 | 222 | 218 | 218 | 217 | 211 | 210 |
| 7 | 222 | 224 | 219 | 221 | 219 | 219 | 213 | 202 |
| 8 | 220 | 221 | 221 | 220 | 220 | 214 | 212 | 201 |

*Table 3. Value of DCT coefficients in subblock (47, 39)*

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 1.742125 | 0.026497 | −0.00791 | 0.001626 | −0.00213 | 0.002284 | −0.00308 | 0.001961 |
| 2 | 0.006479 | −0.01078 | 0.007188 | −0.00681 | 0.005884 | 0.002945 | 0.00139 | 0.00083 |
| 3 | −0.00066 | 0.000533 | −0.00721 | 0.002397 | 0.002546 | 0.004778 | −2.8E−05 | 0.000525 |
| 4 | −0.00084 | −0.00105 | −0.00086 | −0.00084 | 0.002695 | −0.00046 | 0.002963 | 0.000518 |
| 5 | −0.00313 | 0.000609 | 0.000147 | −0.00168 | 0.002625 | 3.5E−05 | −0.00128 | 0.003985 |
| 6 | 0.001678 | 0.002373 | 0.000715 | 0.003409 | 0.00036 | 0.00064 | −0.00122 | −0.00074 |
| 7 | 0.000681 | 0.001202 | 0.001972 | −0.00063 | 0.001246 | 0.001165 | 0.001456 | 0.001745 |
| 8 | 0.000664 | 0.001202 | 0.000994 | 0.001249 | −0.00041 | 0.002797 | −0.00142 | −2.5E−05 |

neural network. For this training, the number of hidden neurons of neural network is set to 10.

Back propagation learning process is repeated until the output values converge to a learning threshold. For this learning, the threshold value for the learning is set to 0.1. This means if the input values are trained to output 1, training stops if output unit outputs a value more than or equal

*Table 4. Values of the DCT coefficients for all 32 subblocks*

| | value1 | value2 | value3 | value4 | value5 | value6 | value7 | value8 |
|---|---|---|---|---|---|---|---|---|
| pattern1 | 1.046 | −0.01417 | 0.005203 | 0.005085 | 0.00225 | 0.006173 | −0.00045 | −0.00109 |
| pattern2 | 1.90625 | 0.001313 | 0.001914 | −0.00166 | 0.00125 | 0.000207 | −0.00091 | −0.00136 |
| pattern3 | 1.737625 | −0.00106 | 0.001597 | −5.2E−05 | −0.00188 | 0.002027 | −0.00035 | −0.00192 |
| pattern4 | 1.257125 | 0.04542 | 0.008758 | 0.001642 | −0.00513 | −0.00034 | −0.00026 | 0.000781 |
| pattern5 | 1.91425 | −0.0027 | 0.001082 | 0.000317 | 0.0005 | 2.36E−05 | −0.00033 | −0.00014 |
| pattern6 | 1.047625 | 0.033841 | −0.02533 | 0.043072 | −0.00163 | −0.00329 | 0.003834 | 0.004879 |
| pattern7 | 1.836875 | 0.000575 | −0.00058 | −0.00099 | 0.001125 | 0.000356 | 0.002076 | 0.000559 |
| pattern8 | 0.89225 | −0.00841 | −0.0159 | −0.00392 | −0.0015 | −0.00533 | −0.00635 | 0.005165 |
| pattern9 | 0.773625 | 0.041021 | 0.028334 | 0.004469 | −0.00063 | −0.00257 | −0.00083 | −0.00342 |
| pattern10 | 1.5535 | 0.022212 | 0.018848 | 0.003559 | −0.00375 | −0.00299 | −0.0006 | −0.00078 |
| pattern11 | 1.47675 | −0.0186 | 0.00353 | 3.42E−06 | 0.00175 | 0.0035 | 0.00247 | 0.001598 |
| pattern12 | 1.839125 | −0.00126 | 0.000808 | 0.001392 | 0.000875 | −0.00161 | 0.001692 | −1.8E−05 |
| pattern13 | 0.990625 | −0.05766 | 0.000265 | 0.00908 | −0.00363 | −0.00211 | −0.00027 | −0.00431 |
| pattern14 | 1.6185 | −0.0325 | −0.00097 | 0.002016 | 0.007 | 0.003119 | −0.00203 | 0.000866 |
| pattern15 | 1.906625 | −0.0051 | −0.00087 | 0.00148 | −0.00263 | −0.00029 | 0.000369 | 0.002401 |
| pattern16 | 0.740875 | 0.14334 | −0.01397 | −0.01949 | −0.00013 | −0.0079 | 0.006716 | −0.00244 |
| pattern17 | 0.852125 | −0.01074 | −0.00097 | −0.0027 | 0.000375 | −0.00178 | 0.000972 | −0.00328 |
| pattern18 | 1.624875 | 0.005121 | 0.016029 | −0.01111 | −0.00263 | −0.00123 | 0.007721 | 0.000218 |
| pattern19 | 0.69225 | 0.052677 | 0.013786 | −0.00015 | −0.00075 | −0.00249 | 0.006715 | 0.001961 |
| pattern20 | 1.518125 | −0.00257 | 0.007749 | 0.003048 | 0.001125 | −0.00082 | 0.000501 | −0.00015 |
| pattern21 | 1.02175 | 0.014577 | −0.03787 | 0.029608 | −0.00675 | 0.001834 | 0.001372 | −0.00152 |
| pattern22 | 1.83775 | 0.001225 | −0.00216 | 0.003583 | −0.002 | 0.001319 | −0.00109 | −0.00163 |
| pattern23 | 1.65075 | −0.0077 | 0.003828 | 0.001929 | −0.00025 | −0.0005 | −0.00183 | 0.002772 |
| pattern24 | 1.616625 | 0.011404 | −0.00435 | 0.003599 | 0.003625 | 0.000409 | −0.0024 | 0.001089 |
| pattern25 | 1.688125 | 0.001867 | −0.00336 | −0.00125 | −0.00163 | 0.000259 | 0.000356 | −0.00038 |
| pattern26 | 0.801125 | −0.01131 | 0.003799 | −0.00825 | 0.000125 | −0.00029 | 0.006451 | 0.001357 |
| pattern27 | 0.761125 | −0.00989 | −0.0026 | −0.00261 | 0.000125 | −0.00042 | −0.00065 | 0.001919 |
| pattern28 | 0.871375 | −0.00438 | 0.067719 | −0.03157 | 0.011125 | −0.00072 | 0.001781 | −0.00634 |
| pattern29 | 1.261875 | 0.005028 | −0.03984 | −0.05209 | −0.01413 | −0.01511 | −0.00041 | −0.00434 |
| pattern30 | 1.43275 | 0.002375 | −0.00365 | 0.003301 | −0.00275 | 0.000992 | 0.001652 | 0.003332 |
| pattern31 | 1.528375 | −0.00235 | 0.002131 | 0.002747 | 0.004125 | 0.002819 | 0.003369 | −0.00022 |
| pattern32 | 1.742125 | −0.01078 | −0.00721 | −0.00084 | 0.002625 | 0.00064 | 0.001456 | −2.5E−05 |

to 0.9, and if the input values are trained to output 0, training stops if output unit outputs a value less than or equal to 0.1. Also, the output threshold in this example is set to 0.5. This means if the output value is larger than 0.5, output signal is set to 1, and if the output value is smaller than 0.5, output signal is set to 0. This training process is repeated for all five neural networks. Eventually, networks are trained to output values for the subblock (47, 39), for example, as Figure 7.

Examples of the output values for each trained network for 32 input values are shown in Figures 8, 9, 10, 11, and 12.

Finally, the network weights of the networks are saved as extraction keys, as well as the coordinates of the selected feature subblocks. Here, network weights between the input layer and hidden layer are named *W*1, and network weights between the hidden layer and output layer are named *W*2. Network weights *W*1 and *W*2 for the five networks of the training are shown in Tables 5, 6, 7, 8, 9 ,and 10. For *W*1, the table has 11*9 values that represent a network weight between the 11 hidden layer units and 9 input layer units. For *W*2, the table has 1*11 values that represent a network weight between the output unit and 11 hidden layer units.

*Figure 7. Converged networks and the flow for output values of subblock (47, 39)*

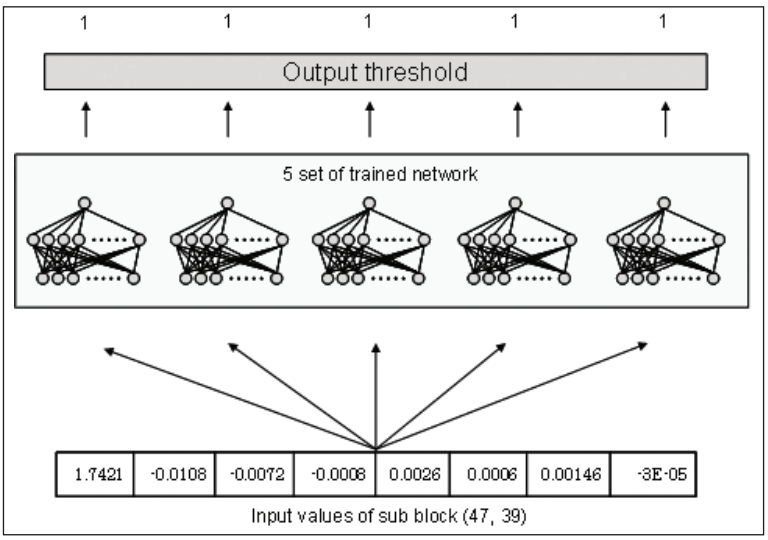



*Figure 8. Output signals of network 1*

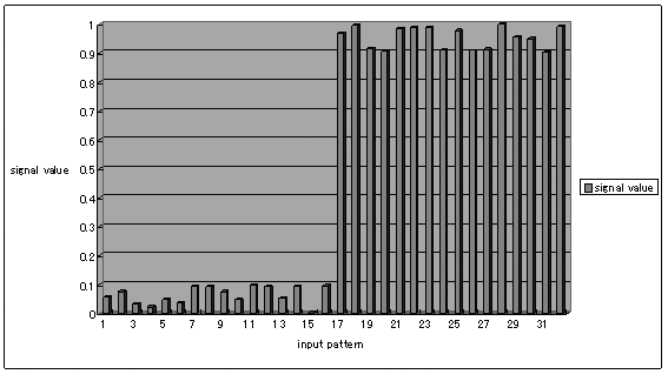


*Figure 9. Output signals of Network 2*

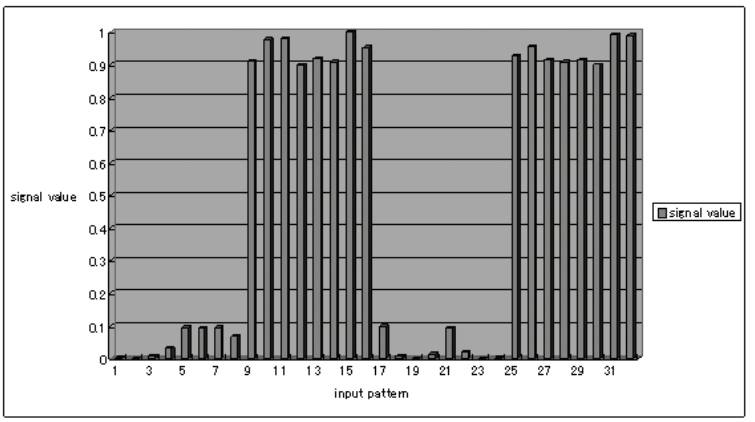



*Figure 10. Output signals of Network 3*

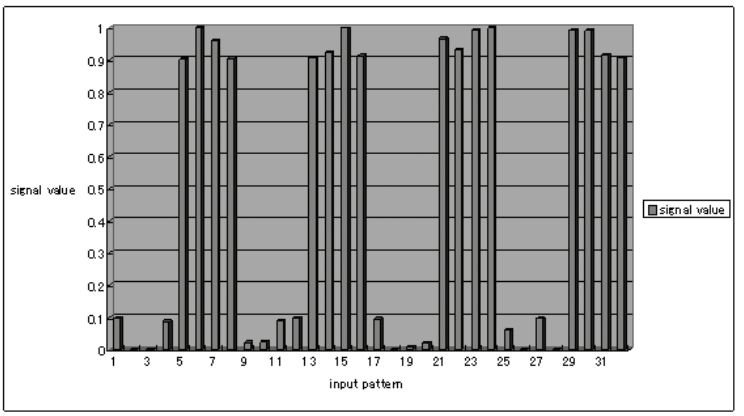



*Figure 11. Output signals of Network 4*

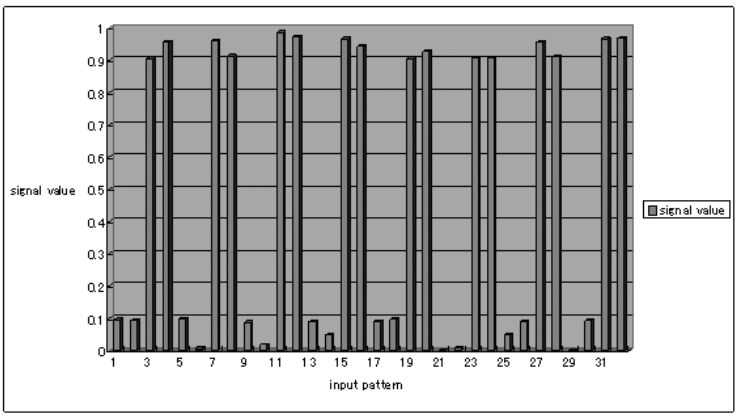


*Figure 12. Output signals of Network 5*

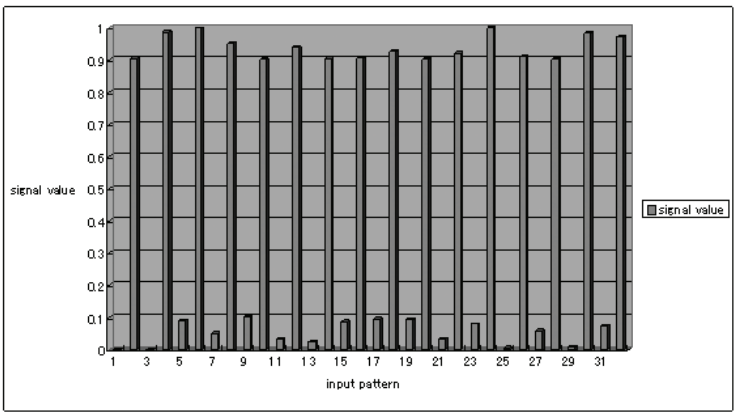



*Table 5. Values of W1 in network 1*

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | −0.545 | 0.1053 | −0.318 | −0.409 | −0.849 | −0.253 | −0.329 | −0.638 | −2.816 | −0.8447 | −2.2851 |
| 2 | −6.243 | 4.2957 | 1.6684 | −0.057 | −2.049 | 5.7934 | −2.131 | 2.4219 | −0.212 | −0.279 | −1.5509 |
| 3 | 1.1761 | −2.775 | −2.683 | 0.2241 | 2.895 | −8.455 | 7.7772 | 0.0238 | 8.9507 | 0.69486 | −0.0214 |
| 4 | −0.838 | 0.9695 | −3.346 | 0.2575 | −0.603 | −2.367 | 1.6415 | −1.375 | −0.17 | 0.78862 | 0.17302 |
| 5 | 2.7196 | 0.7128 | 0.9428 | 0.085 | −0.073 | 2.6079 | 2.3858 | 0.5399 | −2.515 | −0.0892 | 0.04078 |
| 6 | 2.0686 | 0.1655 | 0.5588 | −0.618 | −2.737 | 3.1482 | −4.191 | −0.942 | −4.895 | −0.0161 | −0.7824 |
| 7 | −0.642 | −2.027 | 3.1139 | −0.099 | −0.362 | 7.2715 | 2.698 | 1.7409 | −1.547 | −0.2812 | −0.9968 |
| 8 | −0.072 | −0.405 | −1.32 | −0.546 | −0.4 | −4.601 | −0.374 | 1.0797 | 1.1151 | −0.5249 | −1.394 |
| 9 | −0.415 | −0.317 | 0.2238 | −0.175 | −0.737 | 0.1087 | −0.432 | −0.732 | −1.891 | −0.6687 | −1.307 |

*Table 6. Values of W1 in network 2*

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | −1.367 | 1.8037 | −0.946 | −2.038 | −0.974 | −0.293 | −0.662 | 0.776 | −1.044 | −2.1545 | −1.411 |
| 2 | −0.084 | 9.9794 | 0.4168 | 3.9 | 2.8618 | 1.3901 | −0.426 | 2.0642 | −3.191 | 7.2107 | −2.9311 |
| 3 | 1.0003 | 3.2476 | 0.2048 | 2.8046 | 2.85 | −0.915 | 0.069 | −2.509 | −2.042 | 6.2208 | −1.2299 |
| 4 | −1.126 | −1.968 | −1.147 | 2.8229 | 1.7002 | 1.5579 | −0.148 | 2.0128 | 0.4605 | 6.0481 | 1.7382 |
| 5 | 0.123 | −2.412 | 0.4961 | 0.6458 | 0.0802 | −2.341 | −0.113 | −4.172 | 1.7989 | −1.6866 | −3.881 |
| 6 | 0.5489 | 1.7829 | −1.183 | 4.6794 | 3.1993 | −0.254 | 0.1961 | 0.2275 | 5.3526 | −3.9023 | 2.8114 |
| 7 | −2.184 | −6.115 | 0.1148 | 0.3638 | −0.356 | 0.4263 | −0.416 | −1.566 | 2.8082 | −6.332 | −2.5444 |
| 8 | 0.0054 | −0.448 | 1.0357 | 2.7707 | 2.3143 | 0.6939 | 0.0399 | 1.0644 | 6.0917 | −2.4388 | 0.31353 |
| 9 | −1.015 | 0.9244 | −0.235 | −1.366 | −0.828 | 0.0797 | −0.675 | 0.3011 | −0.596 | −1.6262 | −0.668 |

*Table 7. Values of W1 in network 3*

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | −0.52 | 0.5308 | −1.014 | 0.3395 | 0.102 | −0.991 | −3.071 | −0.549 | −0.597 | −0.5042 | −1.1414 |
| 2 | −0.031 | −0.135 | 0.7059 | 0.8381 | −0.678 | 0.3264 | 1.32 | 2.5973 | −0.817 | −1.4803 | −0.6406 |
| 3 | −0.699 | −2.534 | 1.6019 | 0.9982 | −0.058 | 0.7466 | 10.143 | 1.9797 | 1.3283 | −1.1433 | −3.5568 |
| 4 | 0.2164 | 1.9005 | −4.163 | 2.2266 | −5.457 | −1.904 | −0.588 | 4.3478 | 0.5493 | 1.9031 | 1.2143 |
| 5 | 0.3312 | −1.035 | −3.48 | 0.5962 | −2.432 | −1.026 | 3.1225 | 0.377 | 0.8562 | 0.01517 | 0.9234 |
| 6 | 0.1945 | −1.733 | 8.6322 | −1.513 | 4.644 | 1.1497 | −7.22 | −2.22 | 2.5815 | 0.36396 | −0.1429 |
| 7 | −0.239 | −1.795 | −3.367 | 0.4196 | −0.65 | −0.153 | 1.9698 | 0.0928 | 1.4199 | −0.638 | −0.4532 |
| 8 | −0.004 | −0.189 | 0.7364 | −0.044 | 0.3327 | 0.1784 | −6.282 | 0.9476 | 0.7248 | 0.41934 | 0.33366 |
| 9 | −0.396 | −0.006 | −0.285 | 0.3719 | −0.042 | −0.431 | −2.437 | −0.668 | −0.269 | −0.4198 | −0.4709 |

*Table 8. Values of W1 in network 4*

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | −0.188 | −0.842 | −6.098 | −0.282 | −0.605 | −0.387 | 1.7103 | 0.4583 | 0.8019 | −0.8733 | −1.5495 |
| 2 | −0.391 | −1.071 | 2.6207 | −0.202 | 0.4143 | −0.056 | −2.921 | 0.0196 | −0.21 | −0.7724 | −1.3532 |
| 3 | −0.763 | 1.1549 | 0.2007 | 0.784 | −2E−04 | −0.483 | 3.7897 | 1.9257 | 1.3404 | 1.8556 | −1.1434 |
| 4 | −0.331 | −7.813 | −1.796 | 0.2511 | 0.2833 | −0.8 | −3.725 | 1.8876 | 2.0319 | 1.8928 | −1.8213 |
| 5 | 2.3896 | −4.29 | 4.4608 | 1.3475 | 4.3159 | 1.6742 | 6.0958 | −2.256 | −3.004 | −0.2381 | 0.97661 |
| 6 | −0.477 | −0.485 | −4.867 | −0.465 | −2.008 | −0.885 | 1.166 | 1.297 | 1.5709 | 2.0531 | −0.9452 |
| 7 | 0.9288 | 2.6817 | 7.8704 | 1.4954 | 2.2435 | 2.025 | −1.197 | −0.304 | −0.643 | −1.8457 | 0.62916 |
| 8 | 1.9688 | 1.4654 | 9.6635 | −0.771 | 0.2419 | 2.4045 | 1.2159 | 1.8784 | 2.8279 | −1.3986 | 1.0164 |
| 9 | −0.153 | −1.01 | −4.002 | −0.083 | −0.558 | 0.011 | 1.0589 | 0.0689 | 0.7535 | −0.6896 | −0.7693 |

*Table 9. Values of W1 in network 5*

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | −0.44 | −0.135 | −3.037 | −0.515 | 1.164 | −2.161 | −1.541 | 0.6785 | −0.706 | −1.0659 | −3.1528 |
| 2 | −0.327 | 0.9845 | 4.3372 | 0.0297 | 1.9898 | 3.7382 | −6.918 | −4.225 | −1.315 | −2.6792 | 0.26183 |
| 3 | 0.1797 | 0.255 | 2.014 | 0.4605 | 7.1733 | 3.9717 | 0.8953 | 3.2146 | 0.7123 | −0.3898 | −0.9424 |
| 4 | 0.2452 | 0.2194 | −0.129 | 0.6104 | −6.218 | 1.348 | 1.4185 | −5.954 | 1.9689 | −0.0146 | −1.356 |
| 5 | −1.256 | −5.075 | 4.3461 | −3.793 | −3.498 | 1.6737 | −3.317 | 5.1555 | −3.668 | −1.85 | −0.0355 |
| 6 | 1.1968 | 0.9939 | −1.903 | 0.3297 | 2.1827 | −3.403 | 5.0701 | −4.061 | 1.727 | 1.4227 | −1.1108 |
| 7 | −0.749 | −2.344 | −0.356 | −1.375 | 3.3366 | −0.005 | −1.592 | 3.7412 | −1.566 | −0.1358 | −0.717 |
| 8 | 0.3522 | 1.3888 | 5.1627 | 0.8912 | −0.982 | −0.315 | 1.9615 | −5.07 | 2.0683 | −1.0633 | −1.0107 |
| 9 | −0.337 | −0.493 | −1.764 | −0.253 | 0.7348 | −1.286 | −1.318 | 0.2298 | −0.349 | −0.8304 | −1.9413 |

*Table 10. Values of W2 for all networks*

| | Network 1 | Network 2 | Network 3 | Network 4 | Network 5 |
|---|---|---|---|---|---|
| 1 | 4.7438 | −0.91436 | 0.58293 | −2.6139 | 1.3252 |
| 2 | −3.95 | −9.189 | 3.1252 | −7.7125 | 4.735 |
| 3 | 0.7001 | −1.1951 | −7.7107 | 9.9937 | 5.3579 |
| 4 | 0.41432 | 7.5362 | 2.5274 | −2.0389 | 3.1819 |
| 5 | 3.1558 | 5.1853 | −5.2972 | −4.7375 | −12.4253 |
| 6 | 8.6444 | 2.3507 | −1.2873 | −2.799 | 2.7128 |
| 7 | −9.0507 | 0.41996 | −11.5006 | 8.7345 | −5.4653 |
| 8 | −2.5667 | 5.9921 | 4.72 | −1.7085 | 9.1684 |
| 9 | 9.1814 | −9.8637 | −2.1901 | −2.6973 | 4.2928 |
| 10 | 0.8907 | −8.8103 | 1.588 | 3.5318 | −1.5109 |
| 11 | 1.115 | 3.0858 | 1.806 | −0.62734 | −1.3119 |

## Extraction of Watermark Signal

Now we examine the extraction process in detail. We take the target content, Lena image, and process a frequency transform using DCT. We load the coordinates of selected feature subblocks and the network weights to build a neural network for extraction of the watermark signals. We simply feed forward the network with the input values, the DCT coefficients values that are induced from the selected feature subblocks. If the target content is the same image as the embedded image, with having proper input values and network weights, network will output the same watermark signals.

*Figure 13. High pass filtered Lena image*

## Experiments and Future Works

### Experiment 1. Extraction of Watermark signals from high pass Filtered Lena Image

In this experiment, we will examine if we can retrieve a watermark signal from a graphically changed image. Here, we chose high pass filter as the alteration method. The high pass filtered Lena image is shown in Figure 13.

First, we load the coordinates of the feature subblocks and the network weight of the trained network for the original Lena image. Then, we retrieve input values for each pattern from the high pass filtered Lena image. Input values are

*Table 11. Input values of the high pass filtered Lena image*

| | value1 | value2 | value3 | value4 | value5 | value6 | value7 | value8 |
|---|---|---|---|---|---|---|---|---|
| pattern1 | 1.00888 | −0.0139 | 0.004637 | 0.00496 | 0.00263 | 0.00595 | −0.0001 | −0.001 |
| pattern2 | 1.14925 | 0.00056 | 0.001914 | −0.0016 | 0.00125 | −2E−06 | −0.0009 | −0.0015 |
| pattern3 | 1.12463 | −0.0015 | 0.001524 | −0.0001 | −0.0024 | 0.00246 | −0.0008 | −0.0018 |
| pattern4 | 0.79388 | 0.04599 | 0.008206 | 0.0019 | −0.0051 | −0.0003 | −0.0005 | 0.00089 |
| pattern5 | 1.14588 | −0.0045 | 0.000869 | 0.00026 | 0.00063 | 3.8E−05 | −0.0004 | 0.0002 |
| pattern6 | 1.14463 | 0.03505 | −0.02606 | 0.04334 | −0.0009 | −0.0036 | 0.00381 | 0.0047 |
| pattern7 | 1.13475 | −7E−05 | −0.00079 | −0.0011 | 0.001 | 0.00055 | 0.00204 | 0.0006 |
| pattern8 | 0.81063 | −0.0077 | −0.01581 | −0.0041 | −0.0014 | −0.0055 | −0.0064 | 0.00528 |
| pattern9 | 0.98075 | 0.041 | 0.028526 | 0.00453 | −0.0008 | −0.0031 | −0.0015 | −0.0035 |
| pattern10 | 1.07588 | 0.02589 | 0.019186 | 0.0032 | −0.0036 | −0.0031 | −0.0004 | −0.001 |
| pattern11 | 1.20313 | −0.0173 | 0.003244 | 0.00017 | 0.00213 | 0.00398 | 0.00201 | 0.00167 |
| pattern12 | 1.16463 | −0.002 | 0.00111 | 0.0012 | 0.00113 | −0.0017 | 0.00164 | 3.8E−05 |
| pattern13 | 0.91713 | −0.054 | 0.000265 | 0.00905 | −0.0036 | −0.0021 | −0.0003 | −0.0045 |
| pattern14 | 1.28488 | −0.0311 | −0.00076 | 0.00165 | 0.00688 | 0.00353 | −0.002 | 0.00089 |
| pattern15 | 1.17988 | −0.0042 | −0.00069 | 0.00124 | −0.0029 | −0.0003 | 0.00019 | 0.00268 |
| pattern16 | 0.7115 | 0.14118 | −0.01411 | −0.0195 | −0.0005 | −0.0081 | 0.00711 | −0.0026 |
| pattern17 | 1.0425 | −0.0109 | −0.00099 | −0.0023 | 0 | −0.0023 | 0.00149 | −0.0036 |
| pattern18 | 1.18663 | 0.00531 | 0.016029 | −0.0108 | −0.0029 | −0.0013 | 0.00772 | 0.00028 |
| pattern19 | 0.628 | 0.0505 | 0.013859 | −0.0004 | −0.0005 | −0.0024 | 0.00714 | 0.00182 |
| pattern20 | 0.97938 | −0.0045 | 0.008103 | 0.00279 | 0.00138 | −0.0011 | 0.00015 | −0.0002 |
| pattern21 | 1.17125 | 0.01574 | −0.03805 | 0.02973 | −0.0063 | 0.00236 | 0.00155 | −0.0013 |
| pattern22 | 1.11525 | 0.00066 | −0.00203 | 0.00333 | −0.0018 | 0.00125 | −0.001 | −0.0017 |
| pattern23 | 1.1555 | −0.0054 | 0.003828 | 0.00184 | −0.0005 | −0.0006 | −0.0018 | 0.00315 |
| pattern24 | 1.01425 | 0.01084 | −0.00426 | 0.00342 | 0.0035 | 0.00052 | −0.0025 | 0.00122 |
| pattern25 | 1.045 | 0.00216 | −0.00332 | −0.0014 | −0.0018 | 3.4E−05 | 0.00057 | −0.0003 |
| pattern26 | 1.005 | −0.0147 | 0.003732 | −0.0089 | 0 | 2.4E−05 | 0.00727 | 0.0016 |
| pattern27 | 0.985 | −0.0104 | −0.00329 | −0.003 | 0.00025 | −8E−05 | −0.0005 | 0.00196 |
| pattern28 | 0.9915 | −0.0058 | 0.067984 | −0.0315 | 0.011 | −0.0004 | 0.00152 | −0.0063 |
| pattern29 | 1.2675 | 0.00346 | −0.03987 | −0.0523 | −0.014 | −0.0158 | −0.0006 | −0.0044 |
| pattern30 | 1.18825 | 0.00271 | −0.00378 | 0.00245 | −0.0025 | 0.00106 | 0.00153 | 0.00328 |
| pattern31 | 1.00875 | −0.0034 | 0.00222 | 0.00269 | 0.00375 | 0.00291 | 0.00328 | −0.0002 |
| pattern32 | 1.25938 | −0.0087 | −0.00721 | −0.0011 | 0.00238 | 0.00085 | 0.00146 | −4E−05 |

induced using the coordinates of selected feature subblocks, and values are taken diagonally from DCT coefficients. Notice that compared with the input values of the selected feature blocks of original Lena image, the input values retrieved from high pass filtered Lena image are being quite changed, as shown in Table 11.

Feed forward propagation of these input values to the neural network with network weights of the trained network, we get output signals as shown in Figure 14, 15, 16, 17, and 18.

As you can see, the output signals for high pass filtered Lena image are slightly different compared to the output signals for the original image, but with the same output threshold of 0.5 being used as in the learning process, we were able to retrieve the same watermark signals for all 32 sets of input patterns from high pass filtered Lena image.

*Figure 14. Output signals of network 1 (high pass filtered)*

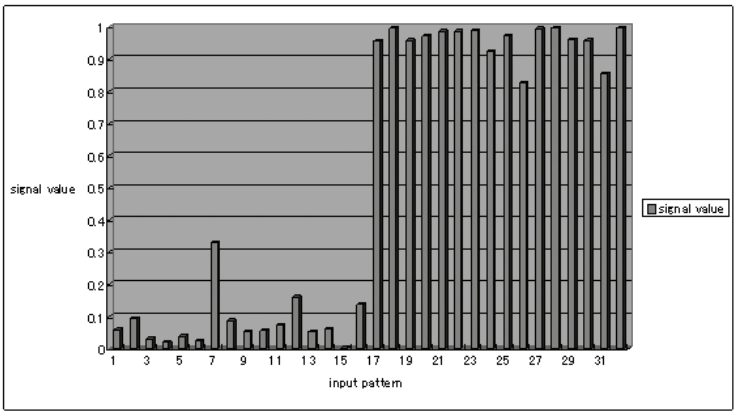



*Figure 15. Output signals of network 2 (high pass filtered)*

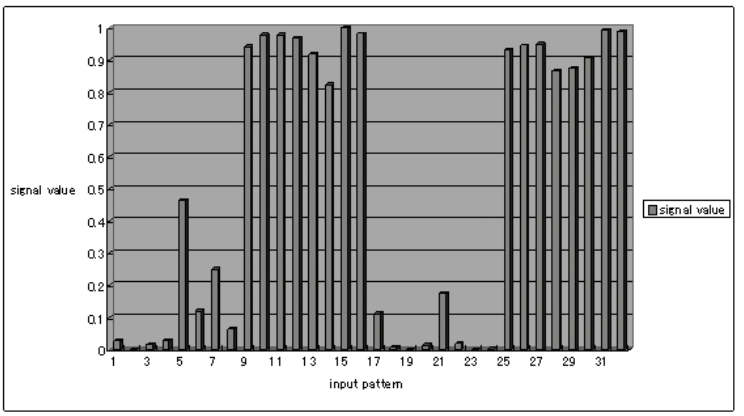


*Figure 16. Output signals of network 3 (high pass filtered)*

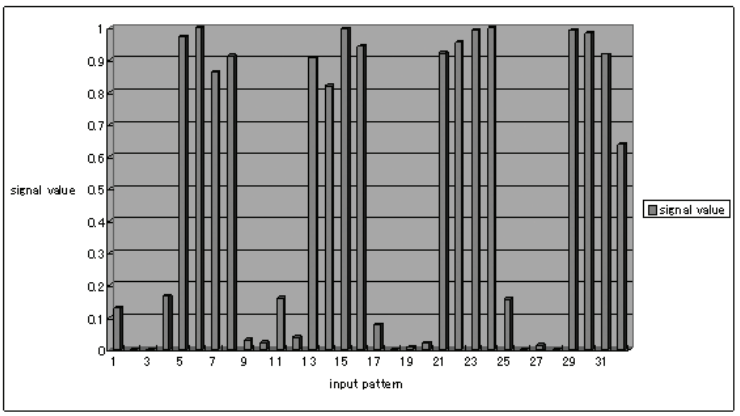



*Figure 17. Output signals of network 4 (high pass filtered)*

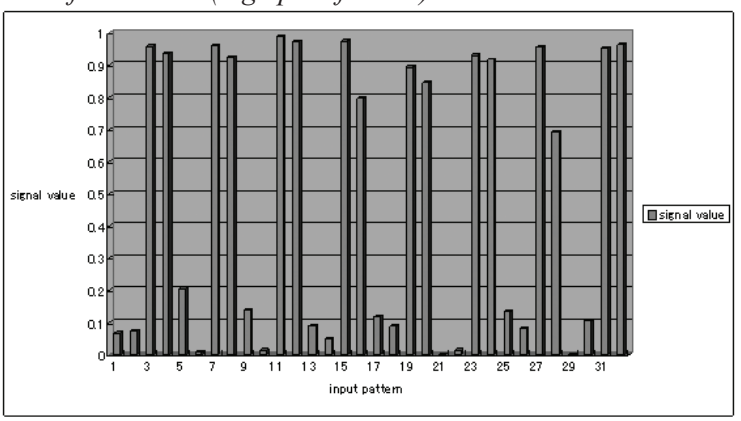
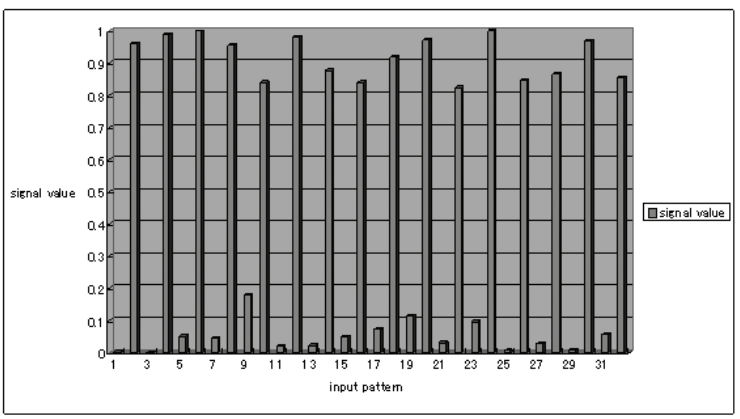



*Figure 18. Output signals of network 5 (high pass filtered)*



## Experiment 2. Changing the Values of DCT Coefficients Directly

In this experiment, we will directly alter the DCT coefficients values of the pixels after the frequency transform is being processed. We change the value of DCT coefficients to 0 for the values lower than the boundary value that increase from 0 to 2 with the step of 0.1. In this experiment, we examined the percentage of proper signal values for the neural network with three different parameter settings as the boundary changes:

a. Number of hidden neurons = 10 and value of learning threshold = 0.05
b. Number of hidden neurons = 10 and value of learning threshold = 0.1
c. Number of hidden neurons = 20 and value of learning threshold = 0.1

There was not much notable difference in the result for (a), (b), and (c), so we can say that the learning threshold and the number of neurons in hidden layer does not affect vastly the output signals. However, the calculation cost and the

*Figure 19. (a) Percentage of output units correctly outputting signals after the alteration of image data (when learning threshold is 0.05 and number of hidden neuron=10)*

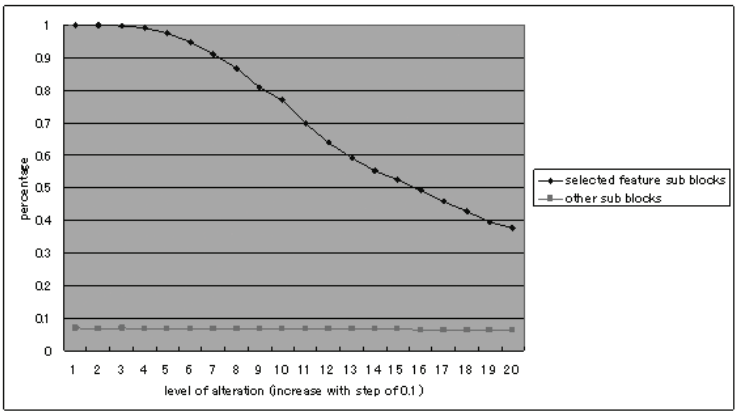



*Figure 20. (b) Percentage of output units correctly outputting signals after the alteration of image data (when learning threshold is 0.1 and number of hidden neuron=10)*

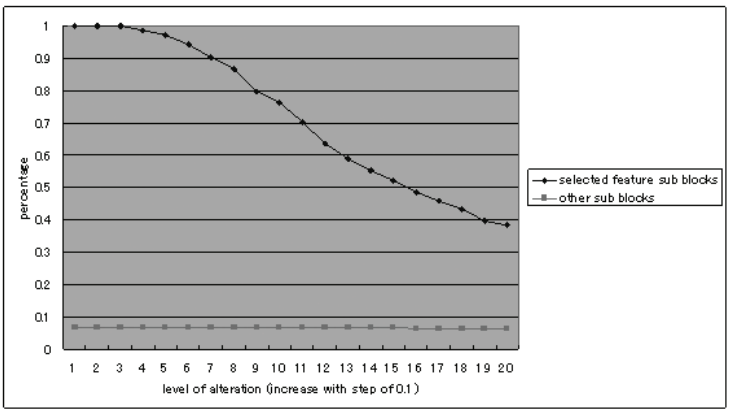
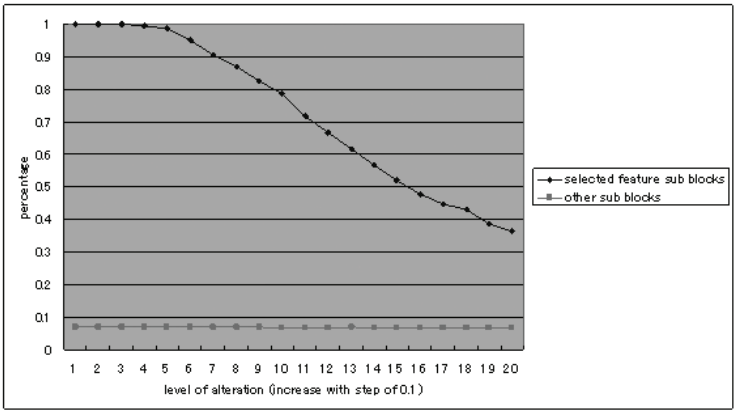


*Figure 21. (c) Percentage of output units correctly outputting signals after the alteration of image data (when learning threshold is 0.1 and number of hidden neuron=20)*




convergence time was higher when more hidden layer neurons were used and lower the learning threshold was predefined.

When we look at the results of this experiment more in detail, we found that once some patterns failed to output the proper watermark signals, some of those patterns succeeded to output proper watermark signals as the alteration process advanced. This characteristic implies, in some cases, that complex connectivity of neural network and other supplement neurons make up for the damaged input neuron to output the proper signals. In table 12, we show the transition of output signals values as boundary values increase by 0.2 for all 32 patterns. You can observe a recovery of output signals in patterns 16, 24, 26, and 28 in this experiment.

## DISCUSSION

In this section, we discuss the contribution and limitations of the proposed method.

The proposed method has shown the possibility of an information-hiding scheme without embedding any data into the target content. With the perspective of information security, this method showed the possibility for the applications for digital watermark and steganography. Using the proposed method for digital watermark, it has both characteristics of robust and fragile watermark. This characteristic is determined by the quality of preprocessing for retrieving of the input values and parameter adjustments for learning process. Meanwhile, because proposed watermark extraction method relies on the position of the feature subblocks, it is weak to geometric attacks like shrinking, expanding, and rotation of the image. This problem will be considered as future works of this proposed method. In experiment 2, notice that the percentage of output signals by subblocks other than the selected feature subblocks is stable. The understanding of this feature will be also considered as future work.

## CONCLUSION

In this chapter, we have proposed an information hiding technique without embedding any data into the target content being employed. This characteristic is useful when the user does not

*Table 12. Output values for all 32 patterns*

| | 0.2 | 0.4 | 0.6 | 0.8 | 1.0 | 1.2 | 1.4 | 1.6 | 1.8 | 2.0 |
|---|---|---|---|---|---|---|---|---|---|---|
| pattern1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| pattern2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| pattern3 | 3 | 3 | 3 | 3 | 11 | 11 | 11 | 11 | 11 | 11 |
| pattern4 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 9 | 9 | 9 |
| pattern5 | 5 | 5 | 5 | 5 | 9 | 9 | 13 | 13 | 13 | 26 |
| pattern6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| pattern7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
| pattern8 | 8 | 8 | 8 | 20 | 20 | 20 | 20 | 27 | 27 | 27 |
| pattern9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 5 | 5 |
| pattern10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 9 | 9 | 9 |
| pattern11 | 11 | 11 | 11 | 11 | 11 | 11 | 3 | 3 | 3 | 3 |
| pattern12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 |
| pattern13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 14 |
| pattern14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 |
| pattern15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 11 |
| pattern16 | 16 | 16 | 16 | 16 | 16 | 16 | 32 | 16 | 16 | 16 |
| pattern17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 |
| pattern18 | 18 | 18 | 18 | 18 | 20 | 20 | 20 | 20 | 8 | 8 |
| pattern19 | 19 | 19 | 19 | 19 | 19 | 19 | 27 | 27 | 27 | 27 |
| pattern20 | 20 | 20 | 20 | 20 | 20 | 20 | 16 | 16 | 16 | 32 |
| pattern21 | 21 | 21 | 21 | 21 | 21 | 21 | 21 | 21 | 21 | 21 |
| pattern22 | 22 | 22 | 22 | 22 | 22 | 22 | 22 | 22 | 22 | 22 |
| pattern23 | 23 | 23 | 23 | 23 | 23 | 7 | 7 | 16 | 16 | 16 |
| pattern24 | 24 | 24 | 24 | 24 | 32 | 24 | 24 | 24 | 24 | 24 |
| pattern25 | 25 | 25 | 25 | 25 | 25 | 25 | 25 | 25 | 25 | 25 |
| pattern26 | 26 | 26 | 26 | 28 | 28 | 28 | 28 | 26 | 26 | 26 |
| pattern27 | 27 | 27 | 27 | 27 | 27 | 27 | 27 | 27 | 27 | 27 |
| pattern28 | 28 | 28 | 28 | 28 | 4 | 3 | 12 | 12 | 28 | 28 |
| pattern29 | 29 | 29 | 29 | 29 | 29 | 29 | 29 | 29 | 29 | 29 |
| pattern30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 27 |
| pattern31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 27 |
| pattern32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 24 | 24 |

want to damage the content, but wishes to protect the intellectual property rights.

The proposed method uses multilayered perceptron neural network model for classifying the input patterns to the corresponding watermark signals. For input values, we used DCT coefficients, but proposed method does not limit the frequency transformation method to DCT alone. Other frequency transformations, such as DFT and DWT, can be used to employ the watermark scheme.

In the experiment, we showed the robustness of this scheme to a high pass filter alteration. Also, for the experiment with direct change of DCT coefficients, watermark signals were not completely lost, but as the degree of alteration becomes stronger, watermark signals have failed to output proper signals and some have recovered. This implies that this scheme has both robust and fragile characteristics, and can be conditioned with the parameter adjustments. The proposed method cultivated the damageless watermark scheme, and we hope more research will be accelerated in the areas of information hiding and information security.

## REFERENCES

Aihara, K. (1990). Chaotic neural networks. In H. Kawakami (Ed.), Bifurcation phenomena in nonlinear system and theory of dynamical systems. *Advanced Series on Dynamical Systems, 8,* 143-161.

Ando, R., & Takefuji, Y. (2003). Location-driven watermark extraction using supervised learning on frequency domain. *WSEAS TRANSACTIONS ON COMPUTERS, 2*(1), 163-169.

Baumgarte, F., Ferekidis, C., & Fuchs, H. (1995). A nonlinear psychoacoustic model applied to the ISO MPEG Layer 3 Coder. Preprint 4087. *99th AES Convention*.

Bender, W., Gruhl, D., & Morimoto, H. (1995). Techniques for data hiding. *Proceedings of SPIE*, *2020,* 2420-2440.

Boney, L., Tewfik, A. H., & Hamdy, K. N. (1996). Digital watermarks for audio signals. *IEEE*

*Proceedings of the International Conference on Multimedia Computing and Systems*, 473-480.

Brassil, J., Low, S., Maxemchuk, N. F., & O'Gorman, L. (1994). Electric marking and identification techniques to discourage document copying. *Proceedings of IEEE INFOCOM'94*, *3,* 1278-1287.

Brassil, J., & O'Gorman, L. (1996). Watermarking document images with bounding box expansion. *Proceedings of the First International Information Hiding Workshop*, 1174, 227-235.

Broomhead, D., & Lowe, D. (1988). Multivariable functional interpolation and adaptive networks. *Complex Systems*, *2,* 321-355.

Chang, C. Y., & Su, S. J. (2005). Apply the counterpropagation neural network to digital image copyright authentication. *2005 9th International Workshop on Cellular Neural Networks and Their Applications IEEE,* 110-113.

Cox, I. J., Kilian, J., Leightont, T., & Shamoon, T. (1997). Secure spread spectrum watermarking for images, audio and video. *IEEE Transactions on Image Processing*, 6(2), 1673-1687.

Cox, I. J., Miller, M. L., & Muttoo, S. K. (2002). Digital watermarking. Morgan Kaufmann Pub.

Davis, K. J., & Najarian, K. Maximizing strength of digital watermarks using neural networks. (2001). *IEEE Proceedings of International Joint Conference on Neural Neworks*, *4*, 2893-2898.

Delaigle, J. F., De Vleeschouwer, C., & Macq, B. (1998). Watermarking algorithm based on a human visual model. *Signal Processing, 66*, 319-335.

Echizen, I., Yoshiura, H., Arai, T., Himura, H., & Takeuchi, T. (1999). General quality maintenance module for motion picture watermarking. *IEEE Transaction on Consumer Electronics*, *45*, 1150-1158.

Eckhorn, R., Reitboeck, H. J., Arndt, M., & Dicke, P. (1990). Feature linking via synchronization among distributed assemblies: simulations of results from Cat Visual Cortex. *Neural Computing*, *2*, 293-307.

Elman, J. (1994). Finding structure in time. *Cognitive Science*, *14,* 179-211.

Gruhl, D., Lu, A., & Bender, W. (1996). Echo hiding. *Proceedings of the First International Information Hiding Workshop*, *1174,* 295-316.

Hong, F., Shi, L., & Luo, T. (2004). A semi-fragile watermarking scheme based on neural network. *IEEE Proceedings of the Third International Conference on Machine Learning and Cybernetics*, *6*, 3536-3541.

Hopfield, J. J., & Tank, D. W. (1985). Neural computation of decisions in optimization problems. *Biological Cybernetics*, *52*, 141-152.

Johnson, J. L. (1994). Pulse-coupled neural nets: Translation, rotation, scale, distortion and intensity signal invariances for images. *Applied Optics*, *33*(26), 6239-6253.

Kahn, D. (1996). The history of steganography. *Lecture Notes in Computer Science*, *1174*, 1-5. Information Hiding, Springer-Verlag.

Katzenbeisser, S, & Petitcolas, F. A. P. (2000). Information hiding techniques for steganography and digital watermarking. Artech House.

Kim, Y. C., Byeong, C., & Choi, C. (2002). Two-step detection algorithm in a HVS-based blind watermarking on still images. *Revised Papers of Digital Watermarking First International Workshop IWDW 2002*, *2613*, 235-248.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*,*43*, 59-63.

Kohonen, T. (1995). *Self-organizing maps*. Springer-Verlag.

Liu, Q., & Jiang, X. (2005). Design and realization of a meaningful digital watermarking algorithm based on RBF neural network. *IEEE International Conference on Neural Networks and Brain*, 1, 214-218.

Lou, D. C., Liu, J. L., & Hu, M. C. (2003). Adaptive digital watermarking using neural network technique. *Proceedings of IEEE 37th Annual 2003 International Carnahan Conference on Security Technology*, 325-332.

Maxwell, T., Giles, C., Lee, T. C., & Chen, H. H. (1986). Nonlinear dynamics of artificial neural systems. *AIP Conference Proceedings, 151*, 299-304.

McCulloch, W. S., & Pitts, W. H. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*,5, 115-133.

Moody, J. E., & Darken, C. (1989). Fast learning in networks of locally-tuned processing units. *Neural Computation*,1, 281-294.

ORuanaidh, J. J. K., & Pun, T. (1998). Rotation, scale and translation invariant spread spectrum digital image watermarking. *Signal Processing*. 66, 303-317.

Poggio, T., & Girosi, F. (1990). Regularization algorithms for learning that are equivalent to multilayer networks. *Science*,247(4945), 978-982.

Pu, Y., Liao, K., Zhou, J., & Zhang, N. (2004). A public adaptive watermark algorithm for color images based on principal component analysis of generalized hebb. *IEEE Proceedings of International Conference on Information Acquisition*, 484-488.

Ramos, M. G.., & Hemami, S. S. (1997). Psychovisually-based multiresolution image segmentation. *Proceedings of International Conference on Image Processing*, *3*, 66-69.

Reither, M. K., & Rubin, A. D. (1998). Crowds anonymity for Web transactions. *ACM Transaction On Information and System Security*, *1*(1), 66-92.

Rosenblatt, F. (1985). The perceptron: Probabilistic model for information storage and organization in the brain. *Psychology Review*, *65,* 386-408.

Rummelhart, D. E., McClelland, J. L,. & the PDP Research Group. (1986). *Parallel Distributed Processing*, *1.* The MIT Press.

Swanson, M. D., Zhu, B., & Tewfil, A. H. (1996). Transparent robust image watermarking. *Proceedings of International Conference on Image Processing*, 3, 211-214.

Takefuji, Y., Lee, K., & Aiso, H. (1992). An artificial maximum neural network: A winner-take-all neuron model forcing the state of the system in a solution domain. *Biological Cybernetics*, *67*, 243-251.

Westen, S. J. P., Lagendijk, R. L., & Biemond, J. (1996). Optimization of JPEG color image coding using a human visual system model. *Proceedings of the SPIE, 2657*, 370-381.

Wolfe, P. J., & Godsill, S. J. (2000). Towards a perceptually optimal spectral amplitude estimator for audio signal enhancement. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, *2*, 821-824.

Zhang, F., & Zhang, H. (2004). Applications of neural network to watermarking capacity. *IEEE International Symposium on Communications and Information Technology 2004*, *1*, 340-343.

Zhang, J., Wang, N. C., & Xiong, F. (2002). Hiding a logo watermark into the multiwavelet domain using neural networks. *Proceedings of 14th IEEE*

*International Conference on Tools with Artificial Intelligence*, 477-482.

Zhang, X., & Zhang, F. (2005). A blind watermarking algorithm based on neural network. *ICNN&B '05 International Conference on Neural Networks and Brain IEEE, 2*, 1073-1076.

Zhang, Z. M., Li, R. Y., & Wang, L. (2003). Adaptive watermark scheme with RBF neural networks. *Proceedings of the 2003 International Conference on Neural Networks and Signal Processing IEEE,* 2, 1517-1520.