IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-6, June 2018 E-ISSN: 2347-2693

High Capacity PVD Steganography Using Back Propagation Artificial Neural Network

Jyoti Pandey[1*], Kamaldeep Joshi [2], Mohit Jangra [3] ,Tanu Garg [4] , Sangeeta [5] , Parth kaushik [6]

[1*]Computer science & Engg., UIET, Maharshi Dayanand University, Rohtak, India
[2] Computer science & Engg., UIET, Maharshi Dayanand University, Rohtak, India
[3] Computer science & Engg., UIET, Maharshi Dayanand University, Rohtak, India
[4] Computer science & Engg., UIET, Maharshi Dayanand University, Rohtak, India
[5] Computer science & Engg., UIET, Maharshi Dayanand University, Rohtak, India
[6] Computer science & Engg., UIET, Maharshi Dayanand University, Rohtak, India

[*]Corresponding Author: jyotipandey2004@gmail.com, Tel.: +91 8708595405

Available online at: www.ijcseonline.org

Accepted: 11/Jun/2018, Published: 30/Jun/2018

***Abstract*—** To represent the image on a screen, several bits is used. Image compression a technique which is used to reduce the number of bits used for representation. Image compression helps in reducing the size of the image which results in less storage space and less cost of the transmission. The image is as compressed as the quality of the image is retained. In this paper, the image is compressed first and then the secret message is hidden in it using Tri-way Pixel Value Difference method. A neural network algorithm called Back Propagation Neural Network Algorithm is used for image compression. The benefit of using back propagation algorithm for using image compression is the vast increase in performance of the system as well as less convergence time for neural network training which not only maintain the quality of the image but also reduce the overall size of the image. This neural network method for image compression has shown a very promising result in image compression. After compression of the image, the secret message is embedded using Tri-way pixel Value Difference method which not only provides imperceptible stego image but also enlarges the capacity of the hidden secret information.

***Keywords*—** Artificial Neural Network, Steganography, Image Compression, Back Propagation Algorithm, Pixel-Value Differencing, Data Hiding.

I. INTRODUCTION

Nowadays, the internet is the primary source of communication. Due to the huge development and use of the internet, there is always a lot of demand for the secure communication. This demand is growing day by day with the increased numbers of users. There is always a concern for secure transmission of data. The privacy is of utmost, especially when transmitted secret information over the internet. To maintain private communication and confidentiality of secret information, cryptography and steganography techniques are used. Cryptography encrypts the information or medium such that it can only be decrypted by using a valid key. Steganography is different than cryptography. Instead of encrypting the secret information, it is embedded in a medium in steganography. This medium can be text, image, audio or video etc. Images are prominently used in steganography. Image steganography has many applications in various fields.

There are various types of image formats available like BMP, PNG, JPEG, GIF, TIFF etc. Each of these file formats has their specific properties and different layout in memory of the computer system. They all support different type of color channel and various type of image resolution. Irrespective of file format, nearly all type of images is used in steganography for cover medium. But JPEG and PNG file formats are widely used in the image steganography. Although any image can be used for hiding secret information, an image with small size is always preferred since it requires low bandwidth and less transmission time. Thus image size should be such that it can be easily transferred over the internet and it does not show any visible sign of steganography. To reduce the size of the image while retaining the quality, image compression is used. Image compression is a technique which reduces the size of the large images. This is done by efficiently coding of data bits of the image. When doing compression, the redundant bits, which are required to represent the image, are reduced. Since the number of bits is reduced, hence less memory space is consumed by image resulting in the small size of the image. The compression helps to reduce the channel capacity which is required to transmit them, thus making their transmission fast. Hence compression can be really beneficial when transmitting large size images.

Image compression is usually divided into two categories: lossy compression and lossless compression. In lossless compression, the compressed image is identical to the original image numerically while in lossy compression scheme, the compressed image is not quite similar to the original image numerically. In lossy compression, quality of the compressed image is degraded as compared to the original image. Usually, slight variation in compressed images due to lossy compression is accepted. This variation should be such that it is invisible to the human eyes. The degradation is caused because all the redundant bits are discarded completely.

In traditional compression methods, an image is divided into several blocks of overlapping pixels and these blocks are used as training set. By encoding these pixel blocks into the weighted set of training and transmitting them for reconstruction of the image, image compression can be achieved. Although compression time is very less in such methods, amount of compression is very less. Also,

data is not completely overlapped in such methods which sometimes result in visible compression. To overcome the limitations of the traditional methods, Artificial Neural Networks (ANN) is used. ANN is a machine learning technique which helps in achieving high compression. It can easily deal with noisy data of the traditional compression schemes. It also has the ability to reduce the number of bits of the compressed image. ANN has the ability to preprocess input patterns to produce simpler patterns with fewer components. ANN provides a better compression data rate as well as sufficient security.

After image compression, steganography scheme is applied which hide the secret information into the cover image. To hide this information, various steganography techniques like Least Significant Bit, Pixel Indicator techniques etc. are used. These techniques are evaluated on the basis of their capacity for hiding data and the quality of the stego image. A steganography technique is considered good if a large amount of data can be hidden into it. Also, stego image should be imperceptible. The steganography technique used here is Tri-way Pixel Value Difference. This technique has high data embedding capacity as well as outstanding imperceptibility of the stego image. In original PVD method, only one direction is used for pixel difference, but in Tri-way PVD, three different directions are considered. This reduces the distortion of the stego image. Also, in this technique, several adaptive rules are also presented.

II. THE PROPOSED METHOD

Phase one: Image Compression Using Back-Propagation Algorithm.

The first phase in our proposed scheme is the image compression. For image compression, Back Propagation Algorithm is used. Back Propagation Algorithm is the most popular neural network algorithm for image compression. Back Propagation algorithm is a type of supervised learning. In supervised learning, inputs and desired outputs are known to the system before training process. So every output instance is known to the system beforehand since it lies between specific ranges. After training process, the error is calculated which is the difference between original image and the compressed image. The goal of the algorithm is to minimize this error i.e. original image and the compressed image should be identical to the human eye. This algorithm has two major processes: the first one is called forward pass and the second one is called the backward pass. This algorithm got its name from the second process of this procedure.

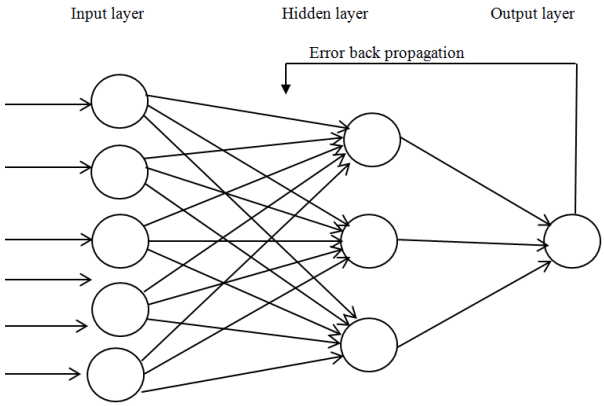


Figure 1: Back-Propagation system

In this algorithm, the difference between the original image and compressed image is minimized. This is done such that the neural network learns to operate on the training data and starts producing output within a specified range. This range can be specified by the user. In this algorithm, training starts with random weights and then, these are adjusted to minimize the error rate. This network has to be designed such that I>J where I is the number of the input layer and output layer neurons while J is the hidden layer neurons. These hidden layer neurons will be used as the output of the compressed image.

Working of the algorithm is as follows:

Each of the input is multiplied by a pre-specified weight which results in inhibition of the input or excitation of the output. Then the weighted sum of the input is calculated.

First total weighted sum is calculated using the formula:

$$M_i=\sum N_i P_{ij}$$

Here, M_i is the total weighted input, N_j is the activity level of the j^{th} unit in the previous layer and P_{ij} is the weight of the connection between the i^{th} and j^{th} unit. To scale the output in between 0 and 1, M_i is passed through a sigmoid function. Then, activity N_j is calculated using some function on the total weighted input. The sigmoid function is similar to the function used here:

$$\mathbf{N_j=\frac{1}{1+e^{-Mj}}}$$

After calculating the output, it is compared with the required output. Now, both outputs are compared and error between them is calculated. Once all the activities of all the units are determined, network computes error X. Error X can be calculated by using following equation:

$$\mathbf{X=\frac{1}{2}\sum_j (N_j-D_j)^2}$$

Where N_i is the activity level of the i^{th} unit in the top layer and D_j is the desired output of the j^{th} unit. In the end, the error is backwardly propagated.

Steps for Neural Network Implementation:

1. Network Definition

The first step in any neural network is the definition of the network architecture. Decision set is connected to the number of neurons in input and output layer. These decisions are connected with the problem. This problem has to be solved through this network. In image compression, the neural network will compress the grayscale image which can be represented by a matrix. In matrix form, the image can be represented using pixel values. The value of each pixel is between 0 and 255. The low value represents the black intensity and high-value represents more white intensity. Usually, the dimension of each block is 8x8 or 64 pixels. These 64 pixels are used for network training which is used as input for the network. The input layer is so constructed that number of neurons in the input layer and output layer are same. The decision of neurons in the hidden layer is not restricted, although their number should be less than the neurons in the input layer. Usually, the number of neurons in the hidden layer is a factor of a block size. So, if 64 neurons are in the input layer, then hidden layer may have 4,8 or 16 neurons. The number of neurons in hidden layer can affect the compression rate of the network.

2. Image pre-processing
In this step, input values are decided for network training. First, the image is divided into 8x8 blocks. Then each block is normalized. Normalization means scaling of the all pixel values in such a way that the input and output values are always in a specified interval. IN case of an image, the value of each pixel is an integer value ranging from 0 to 255. These values are normalized to have them in range of 0 to 1. By doing so, output and input will have same scalarized values . Then each block is linearized using scanning row or columns or by any known method. Linearization means the transformation of the 2D matrix in a one-dimensional array. Then this one dimensional array is fed to the network as the input.

3. Training Pairs preparation
In this step, training pairs are prepared. These pairs are prepared by using normalizing and linearizing the 8x8 blocks. These training pairs are so prepared such that expected output vectors are equal to the input vectors. This step is similar to image pre-processing but improper training pairs are eliminated and only those are selected which are expected to provide output in the specified range.

4. Training of the Network
In a neural network, many algorithms are devised for training. In training, network weights modification is considered. The goal of network training is to minimize error and have similar values of calculated and expected error. When a network achieves its goal at the time of training, we have to remember its weight matrices. This is done in much iteration. The number of iterations should be very less since a large number of iterations can take very long time in compression. These weight matrices are used as parameters to form a completely new image. Then this image is pre-processed and compression is done

5. Compression
After network training, new input and expected outputs are used as input to be fed into network input layer. The expected output is the image that is going to be compressed. Hidden layer output is calculated using previously stored weights. After that, attained values are quantified with 8 bits. These are then stored as a data image. This data image can be further processed to obtain the compressed image. In addition to the compressed image, a 16 bits component is also stored which contains the various image attributes like image dimensions, image format. These attributes can be very helpful in decompressing the image. The compressed image is then compared with the expected output. If there is no visible change in the compressed image, then it is accepted otherwise rejected.

6. Decompression
Decompression is almost the reverse process of the compression. For decompression, The 16 bits stored data of the compressed image is used. These values along with the compressed image are used as the output of the hidden layer. Then, using stored weight matrices, network output is calculated. Operation between hidden layer weight and output layer weight results in the reconstructed image.

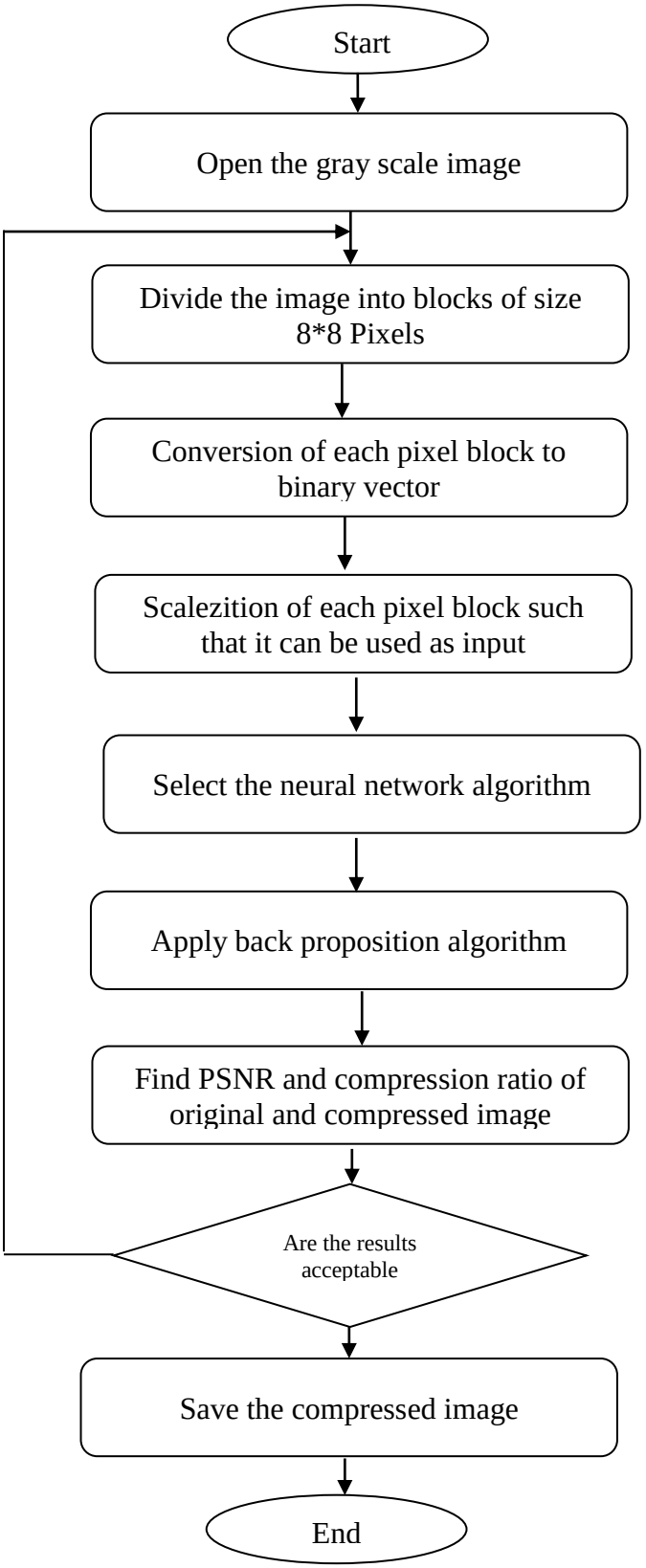


Figure 2: Image Compression Using Multi layered FFBP Algorithm

Algorithm:

1. Initialize the weights to small random values.
2. Select a training vector pair for training set which consists of input and the corresponding output. Use this training pair as the input to the neural network.
3. Next step is the forward phase. In this step, actual output is calculated.
4. This step is the backward phase. It is the most important step of the algorithm. In this step, weights are adjusted according to the error i.e. according to the difference between actual output and desired output. This step is called backward phase because we have to go to previous steps to adjust the weight in such a way that difference is reduced.
5. For all training vectors, repeat step 2.
6. Repeat step 2 until error is small enough to be accepted.

B. Phase 2: Image Steganography Using Tri-way Pixel Value Difference Technique

The second phase of proposed scheme is the Image steganography. In image steganography, a secret message is hidden into a cover image in such a way that changes in the image are not visible to human eyes. There are various image steganography techniques used for hiding information. The capacity and imperceptibility of the steganography technique are the important attributes for any steganography technique. A technique is considered good if it has high capacity and cannot be easily detected.

For our proposed scheme, a variation of Pixel Value Difference (PVD) Technique is used for image steganography. This technique is called Tri-way Pixel Difference Technique. This improves the capacity of original PVD method.

In original PVD method, a grayscale image is into many blocks. These blocks do not overlap with each other. These blocks are composed of two consecutive pixel p_i and p_{i+1}. The difference between these pixels is calculated as d_i by subtracting p_i from p_{i+1}. This difference can range from -255 to 255. To scale the difference, the absolute value is used. Therefore, absolute value of $|d_i|$ is form 0 to 255. . If the value of $|d_i|$ is small, then this block have and value is too high, then this block has very sharp edges. A human eye can easily detect a change in smooth areas in comparison of edge or sharp areas. Therefore, amount of data that can be embedded into edge areas is much larger than smooth areas. A range table T_m is designed in such a way that it contains n continuous ranges where m can be any natural integer up to n. The upper and lower boundary of this range table is U_m and L_m. The width is used to determine the amount of bits that can be embedded into two consecutive pixels. This width w_m can be calculated using formula:

$$w_m=U_m-L_m+1$$

This range table T_m is extremely useful in extraction of the embedded data in the image.

Embedding process in original PVD Method:

1. First, difference d_i is calculated between two consecutive pixels p_i and p_{i+1} of each bloack in cover image.

$$d_i=p_{i+1}-p_i$$

The absolute value of d_i is used in the further calculation. So $|d_i|$ is used.

2. Now, we have to find a range value from the range table. This can be located using $|d_i|$. If R_p is the desired range, then its location j can be calculated as

$$j=\min (U_p-|d_i|)$$

Where U_p is always greater than or equal to $|d_i|$ and p lies between 1 and n. After calculating j, desired range is R_j.

3. Now, using R_j, we can calculate the amount of data that that can be embedded into each of the two consecutive pixels. This data is embedded into bits. This amount of secret data bits s can be estimated using width w_j.

$$s=\text{floor } (\log_2(w_j))$$

4. Now, these secret data bits s is converted into their decimal equivalent b. suppose s was 111100010, then its decimal equivalent will be 482.

5. Now, new difference value $dnew_i$ is calculated. This new value depends on the original value.

If d_i is positive or equal to zero, then:

$$dnew_i=L_j+b$$

or if d_i is negative, then:

$$dnew_i=-(L_i+b)$$

After finding new difference value, d_i is replaced with d'_i

6. Now, the pixel value of the pixel p_i and p_{i+1} is updated. New values $pnew_i$ and $pnew_{i+1}$ will be:

$$pnew_i=p_i-\text{ceil}(df/2)$$

$$pnew_{i+1}=p_{i+1}+\text{floor}(df/2)$$

Where df is the difference between original difference value and new difference value i.e.

$$df=dnew_i-d_i$$

7. Repeat steps 1-6 until all secret data is embedded into cover image and stego-image is obtained.

Extraction in original PVD Method:

For extraction, range table T_m is needed. Following steps are involved in extraction of the secret message from stego-image:

1. Divide the stego image into blocks of pixels.
2. Calculate the difference value d'_i for every consecutive pair of pixels p'_i and p'_{i+1}
3. Similar to embedding process, difference value d'_i is required to locate a suitable R_j in range table.
4. Now lower boundary value L_j is subtracted from the absolute value of d'_i to obtain the value of b'. This b' is the decimal equivalent of the secret bits which is in binary. If there is no change in stego-image at the particular block, then this b' will be equal to b
5. In the final step, this decimal value b' is converted to binary equivalent which contains secret data bits s.Similar to embedding, s will be given by the formula:

$$s=\text{floor } (\log_2(w_j))$$

Tri-way Pixel Value Difference Method:

In original PVD method, only one direction is considered. Since only one vertical edge can be represented by two horizontal pixels, we cannot embed data to its full potential. To increase the embedding capacity, we can use multiple directions instead of a single direction. So in tri-way PVD method, three directions are considered instead of one in original PVD method.

According to PVD method, we can hide information efficiently using a pair of two pixels in a direction. By, using the same concept, we can achieve high efficiency if we consider four pixel pairs in all four directions i.e. in horizontal, vertical and two diagonals. But practically, only three directions can be efficiently used since the value of first and second pixel pairs is affected by modifying the fourth pixel pair. Hence, only three directions are used. Take an example of a 2x2 pixel block consisting of 4 pixels. Like in PVD method, the image is partitioned in the non-overlapping 2x2 pixel blocks.

Consider a 2x2 pixel block as shown in the image. A 2x2 block consists of 4 pixels $p_{(x,y)}$,$p_{(x+1,y)}$, $p_{(x,y+1)}$ and $p_{(x+1,y+1)}$. Here x and y are the horizontal and vertical coordinates of the image. If $p_{(x,y)}$ is the starting point, then $p_{(x+1,y)}$ is the pixel next to it horizontally i.e. it is on the right of the starting point, $p_{(x,y+1)}$ is the pixel next to starting point vertically i.e. it is below the starting point and $p_{(x+1,y+1)}$ is the pixel below the $p_{(x+1,y)}$ i.e. this is the lower right to the starting point. Using these four pixels, we can make three pairs of pixels with each pair consisting of the starting point. These three pairs can be P_0, P_1 and P_2.

$$P_0=(p_{(x,y)},p_{(x+1,y)})$$
$$P_1=(p_{(x,y)},p_{(x,y+1)})$$
$$P_2=(p_{(x,y)},p_{(x+1,y+1)})$$

Every P_i will have a different value P'_i after embedding secret information, where i can be 0,1 or 2. The difference value between two pixels of a pair P_i is denoted by d_i and after embedding, this difference value will be d'_i. Although there can be three different values of starting point after embedding information the only one can be used. These three values are from P_0, P_1, and P_2. To select a starting point p'_i, we can use any value and can use this value as the reference point to offset the other two values. For example, if $p'_{0(x,y)}$ is the reference point, then two difference value d'_1 and d'_2 can be derived from the reference point.

Embedding Algorithm:

1. At first, four difference value $d_{i(x,y)}$ are calculated. These are calculated between four pixel pairs as

$$d_{0(x,y)}=p_{(x+1,y)}-p_{(x,y)}$$
$$d_{1(x,y)}=p_{(x,y+1)}-p_{(x,y)}$$
$$d_{2(x,y)}=p_{(x+1,y+1)}-p_{(x,y)}$$
$$d_{3(x,y)}=p_{(x+1,y+1)}-p_{(x,y+1)}$$

2. Then similar to PVD method, absolute difference value is used to locate s suitable $R_{p,i}$ in the range table i.e $|d_{i(x,y)}|$ is calculated. If $R_{k,i}$ is the desired range, then its location j can be calculated as

$$j=\min(U_p,i-|d_{i(x,y)}|)$$

Here $U_{p,i}$ is the upper boundary value and is always greater than or equal to $|d_{i(x,y)}|$ and p lies between 1 and n. After calculating j, desired range is represented as $R_{j,i}$.

3. Similar to original PVD method, using $R_{j,i}$, we can calculate the amount of data that that can be embedded into each of the two consecutive pixels. This amount of secret data bits s_i can be estimated using width w_j.

$$s_i=\text{floor}(\log_2(w_{j,i}))$$

If branch conditions are satisfied by s_i of P_i, then we can process two pixel pairs by original PVD method, else Triway method is used.

4. Convert secret data bits s_i into decimal equivalent b_i.
5. Now, new difference value $dnew_{i(x,y)}$ is calculated. This new value depends on the original value similar to the original method.

If $d_{i(x,y)}$ is positive or equal to zero, then:

$$dnew_i=L_{j,i}+b_i$$

or if $d_{i(x,y)}$ is negative, then:

$$dnew_i=-(L_{j,i}+b_i)$$

6. Now, update the pixel value of p_k and p_{k+1} using the formula:

$$p'_k=p_k-\text{ceil}(df/2)$$
$$p'_{k+1}=p_{k+1}+\text{floor}(df/2)$$

Here, p_k and p_{k+1} are the two pixels in P_i and df is the difference between new difference value and original difference value i.e

$$df=dnew_n-d_n$$

Until now, the value of p_k and p_{k+1} are changed in order to embed the secret information but now, we have to consider branch conditions too. if branch conditions are not satisfied by s_i, then we go to the next step, else only P'_0 and P'_3 are processed using the original PVD method. Then, we compute the s'_i and again, it is checked whether s'_i satisfy the branch conditions or not. If not, then the pixel value of P'_3 is offset and the block is created and algorithm goes to the last step.

7. Select the optimal reference point having minimum MSE value using the selection rules described in the next section. This select point can be used to offset the other pixel pairs.
8. Construct the new block using all the new pixel pairs which contains all the secret information.

Reference Point selection criteria for Minimum MSE:

A stego-image will have a large amount of distortion if there are different reference points. To, reduce the distortion, optimal reference point should be selected. To select the optimal reference point, we use df which is the difference between original and modified difference value between pixels.

A stego-image will have a large amount of distortion if there are different reference points. To, reduce the distortion, optimal reference point should be selected. To select the optimal reference point, we use df which is the difference between original and modified difference value after embedding between pixels. i.e.

$$df=dnew_i-d_i$$

Following cases are generated:

1. If df_i<-1 or df>1, then optimal pair i_{op} is the pair having largest |df|.
eg. if df_i={-2,-4,-5} and i ∈ {0,1,2} , then i=2

2. If df_i is of the same sign and only one value of df_i lies between -1 and 1, then optimal pair i_{op} will be selected from the other two pairs having smallest value of |df|.
eg. if df_i={3,8,1} and i ∈ {0,1,2}, then i=0

3. If only one df_i is of the different sign than the other, then optimal pair i_{op} is elected from the other two pairs having smallest |df|.
eg. if df_i={5,2,-4} and i ∈ {0,1,2}, then i=1

4. If only one value of df_i lies between -1 and 1 and the other two value are of different signs, then optimal pair i_{op} is the one whose value lies between -1 and 1.
eg. if df_i={3,-2,0} and i ∈ {0,1,2}, then i=2

5. If more than one value lies between -1 and 1, then any pair can be selected as optimal pair i_{op}.
eg. if df_i={2,1,0} and i ∈ {0,1,2}, then i can be 1 or 2

Branch Conditions:
Although this technique has a very large capacity and can be used for embedding a large amount of data, this can still cause distortion in the image which is generated mostly due to the offsetting process, therefore two conditions called branch conditions are devised to avoid offset as much as possible. If branch conditions occur, then we process pixel pairs in one direction using original PVD method.

These conditions are:
1. Embedded_Bit(P_0) is greater than 5 and embedded_Bit (P_1) is greater than 4
2. Embedded_Bit(P_0) is less than 5 and embedded_Bit (P_2) is greater than 6
where embedded_Bit(P_0) represents the total embedding bits along the direction of P_i.

Extraction Algorithm:
1. First, we have to divide the stego-image into non-overlapping 2x2 pixel blocks.This is similar to the embedding process partition.
2. Now, the difference value for each block is calculated as
$d'_{0(x,y)}=p_{(x+1,y)}-p_{(x,y)}$
$d'_{1(x,y)}=p_{(x,y+1)}-p_{(x,y)}$
$d'_{2(x,y)}=p_{(x+1,y+1)}-p_{(x,y)}$
$d'_{3(x,y)}=p_{(x+1,y+1)}-p_{(x,y+1)}$

3. Like embedding phase, absolute difference value $|d'_{i(x,y)}|$ is used to find $R_{p,i}$.
4. Then, the amount of embedded bits s'_i is find using the formula:

$$s_i=\text{floor}(\log_2(w_{j,i}))$$

If branch conditions are satisfied by s_i, then any two independent pixels are selected else three pixel pairs are selected.
5. Now, $L_{j,i}$ is subtracted from the selected $|d'_{i(x,y)}|$ and thus, b' is obtained. Similar to original PVD method, if stego-image is not altered, then b' will be equal to b.
6. Convert b into binary sequence s_i.

Example:
Let's consider a 2x2 pixel block shown below:

130	152
148	135

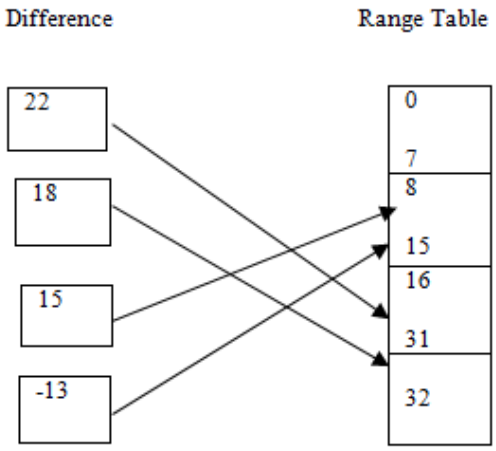
From this, four pixel pairs will be formed as

P_0=(130,152)
P_1=(130,148)
P_2=(130,135)
P_3=(148,135)

The difference of each pixel pair d_i will be

d_0=22
d_1=18
d_2=15
d_3=-13

Now the difference is matched with Range table $R_{j,i}$:



Suppose, the bit stream of the secret message is 1110011001010100.
Width can be calculated as $w_{j,i}=U_p-L_p+1$

Now, the width $w_{j,i}$ for each pixel pair will be
$w(P_0)$=31-16+1=16
$w(p_1)$=31-16+1=16
$w(P_2)$=15-8+1=8
$w(P_3)$=15-8+1=8

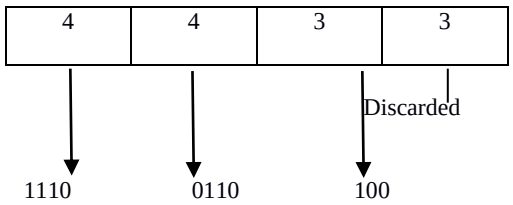
Pixel pair P_3 will be discarded since it satisfies branch conditions.

Now, amount of secret bits which can be embedded into pixel pairs can be calculated as:

$$s_i=\text{floor}(\log_2(w_{j,i}))$$

Therefore,
$s_0=\text{floor}(\log_2(w(P_0)))=\text{floor}(\log_2(16))=4$
$s_1=\text{floor}(\log_2(w(P_1)))=\text{floor}(\log_2(16))=4$
$s_2=\text{floor}(\log_2(w(P_2)))=\text{floor}(\log_2(8))=3$

So,
Data bits for P_0 will be 1110. The decimal equivalent of 1110 is 14. So b_0=14
Data bits for P_1 will be 0110. The decimal equivalent of 0110 is 6. So b_1=6
Data bits for P_2 will be 100. The decimal equivalent of 010 is 2.So b_2=2



Now, the new difference will be $dnew_{i(x,y)}$, which is given by:

$$dnew_i = L_{j,i} + b_i$$

So,
for P_0,$L(P_0)=16$, hence, $dnew(P_0)=16+14=30$
for P_1,$L(P_1)=16$, hence, $dnew(P_1)=16+14=30$
for P_2,$L(P_2)=1$, hence, $dnew(P_2)=8+2=10$

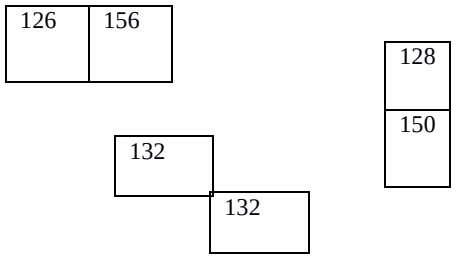
I	**0**	**1**	**2**
$dnew_i$	22	30	8
d_i	18	22	4
Df	15	10	-5

Now, according to optimal reference selection criteria, the optimal value of df will be 4, therefore, i =1

So, new pixel pair can be formed by using formula:

$p'_k = p_k - ceil(df/2)$
$p'_{k+1} = p_{k+1} + floor(df/2)$

By using this for df=4, new pairs of the block will be:



Since, i=1 is optimal reference point, hence the pixel value of this pair will not be changed. For other, offset will be according to the table:

126+(128-126)	156+(128-126)
128	132+(128-128)
150	132+(128-128)

Hence, final pixel block will be:

128	158
150	132

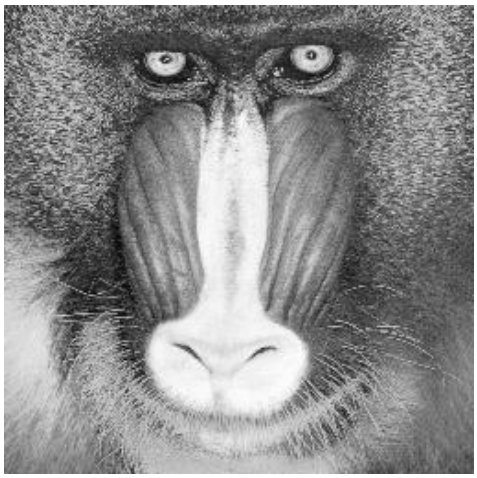
III. RESULT ANALYSIS

Our proposed method improves the capacity of the original method by a significant amount. This increased capacity does not come at the cost of the security. Although the image has been compressed using back propagation network before embedding data into it, the quality of stego-image is still very good and is nearly same as the original image. To check the quality of the stego-image, PSNR value of the original image and stego-image is calculated.

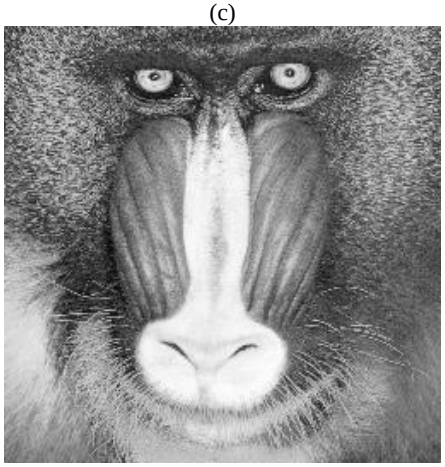
In this method, pseudo-random numbers are used for generation of secret data bits. In the comparison of results with earlier methods, we have used the images with the same dimension as in other methods. Also, results are calculated for images of other dimensions. This is done to show the capacity of images for other resolutions. Example of the cover image and stego-image is shown in the figure. In the table, hiding capacity of 512x512 images, as well as PSNR values with their respective stego images, are compared with original PVD method and TPVD method. For some images, PSNR value may be less than the other methods, but the capacity is much larger than other methods. The difference in images is hardly visible in the case of images having less PSNR than the original methods. This is due to high variance in the pixel value of the image. Therefore, proposed method not only increases the capacity of the image but does so by retaining the same quality as the original.

(a)

(b)



(c)



(d)

Figure 3. Cover image and stego-image. (a) Cover image: Lena (b) Stego-image after embedding secret data with 76224 bytes; PSNR is 38.23 (c) Cover image: Baboon (d) Stego-image after embedding secret data with 87269 bytes; PSNR is 34.12

Table 1: Comparison of Results for the Proposed Method and Existing Methods.

Cover images	PVD Method		TPVD Method		Proposed Method	
	Capacity	PSNR	Capacity	PSNR	Capacity	PSNR
(512×512)	(bytes)	(dB)	(bytes)	(dB)	(bytes)	(dB)
Lena	50960	41.79	75836	38.89	76224	38.23
Baboon	56291	37.90	82407	33.93	87269	34.12
Jet	51243	40.97	76352	38.70	78450	38.02
Peppers	50685	40.97	75579	38.50	76554	37.86

Table 2: Comparison between images of different size using Tri-way Pixel Value Difference Method.

Cover images	Image Size	Proposed Method	
		Capacity (bytes)	PSNR (dB)
Beans	256*256	56363	39.15
Trees	256*256	56854	39.84
House	256*256	56898	39.41
Ca2	320*432	60454	40.17
Peppers	512*512	76554	37.86
Ice Age	444*300	119377	38.22
Penguins	768*1024	693847	40.94
Building	1024*1024	1245466	41.88
Flower	1920*1080	1755681	43.23
Sound Waves	2560*1440	2254454	46.76
Average of 100 images			38.86

IV. CONCLUSION AND FUTURE WORK

In original PVD method, one direction can hide the data efficiently but in TPVD, the capacity of the image is increased very much due to use of three directions. Three different directions can hide much data than one direction. The capacity of TPVD method is increased further by using neural networks. Using Back Propagation Image compression algorithm, the redundant bits were reduced in images which can cause distortion, thus making the image much similar to the original image while still increasing the data hiding capacity of the image. Experimental results have shown that our proposed method not only increase the capacity but does so in a way such that change in the cover image is imperceptible to human vision. Thus this method provides excellent quality of stego-image having higher data hiding capacity. This method is highly robust as compared to other methods and is highly resistant to Steganalysis methods. Also, there is no need of cover image during extraction of data since data can be extracted using only stego-image. Thus, this method has multiple advantages compared to other techniques.

REFERENCES

[1] F. A. P. Petitcolas, R. J. Anderson and M. G. Kuhn, "Information Hiding - a Survey," Proceedings of the IEEE, Vol. 87, pp. 1062–1078, 1999.

[2] W. Bender, D. Gruhl, N. Morimoto, A. Lu, "Techniques for data hiding," IBM Systems Journal Vol. 35 (3–4), pp. 313–336, 1996.

[3] Y. K. Lee, L. H. Chen, "High capacity image steganographic model," IEE Proceedings on Vision, Image and Signal Processing, Vol. 147, No.3, pp. 288-294, 2000.

[4] R.-Z. Wang, C.-F. Lin, and J.-C. Lin, "Image hiding by optimal LSB substitution and genetic algorithm," Pattern Recognition, Vol. 34, pp. 671–683, 2001.

[5] C.-C. Chang, J.-Y. Hsiao, C.-S. Chan, "Finding optimal least-significant-bit substitution in image hiding by dynamic

programming strategy," Pattern Recognition ,Vol. 36, Issue 7, pp. 1583-1595, 2003.

[6] C.-K. Chan, L. M. Cheng, "Hiding data in images by simple LSB substitution," Pattern Recognition Vol. 37, Issue 3, pp. 469-474, 2004.

[7] W.-N. Lie, and L.-C. Chang, "Data hiding in images with adaptive numbers of least significant bits based on the human visual system," IEEE International Conference on Image Processing, Vol. 1, pp. 286–290, 1999.

[8] D.-C. Wu, and W.-H. Tsai, "A steganographic method for images by pixel-value differencing," Pattern Recognition Letters, Vol. 24, pp. 1613–1626, 2003.

[9] H.-C. Wu, N.-I. Wu, C.-S. Tsai, and M.-S. Hwang, "Image steganographic scheme based on pixel-value differencing and LSB replacement methods," IEE Proceedings on Vision, Image and Signal Processing, Vol. 152, No. 5, pp. 611-615, 2005.

[10] S.-L. Li, K.-C. Leung, L.-M. Cheng, and C.-K. Chan,"Data Hiding in Images by Adaptive LSB Substitution Based on the Pixel-Value Differencing," First International Conference on Innovative Computing, Information and Control (ICICIC'06), Vol. 3, pp. 58-61, 2006.

[11] S. Voloshynovskiy, S. Pereira, T. Pun, J. J. Eggers, and J. K. Su, "Attacks on digital watermarks: classification, estimation based attacks, and benchmarks," IEEE Communications Magazine, Vol. 39, Issue 8, 118-126, 2001.

[12] J. Fridrich, M. Goljan, and R. Du, "Detecting LSB steganography in color, and gray-scale images," IEEE multimedia, Vol. 8, Issue 4, pp. 22-28, 2001.

[13] Ko-Chin Changa, Chien-Ping Changa, Ping S. Huangb, and Te-Ming Tua, "A Novel Image Steganographic Method Using Tri-way Pixel-Value Differencing," Journal Of Multimedia, Vol. 3, NO. 2, 2008.

[14] Abbas Razavi, Rutie Adar, Isaac Shenberg, Rafi Retter and Rami Friedlander "VLSI implementation of an image compression algorithm with a new bit rate control capability," Zoran Corporation, 1705 Wyatt Drive, Santa Clara, CA 95054.This paper was published in IEEE international conference ,vol. 5, pp.669-672, 1992.

[15] Ronald A .DeVore, Bjorn Jawerth, and Bradley J.Lucier. "Image compression through wavelet transforms coding," Ieee Transactions on Information Theory, Vol. 38. No .2, 1992.

[16] David Jeff Jackson and Sidney Jwl Hannah "Comparative analysis of image compression technique" Department of Electrical Engineering ,The University of Alabama, Tuscaloosa, AL 35487.This paper appears in system theory 1993,proceeding SSSST'93, twenty fifth edition southeastern symposium, pp. 513-517, 1993.

[17] P.Moravie, H.Essafi, C. Lambertt-Nebout and J-L. Basill. "Real time image compression using SIMD architecture" Centre Spatial de Toulouse 18 Avenue Edouard Belin BP 1421. This paper appears in computer architecture for machine perception, 274-279, 1995.

[18] JJiang "Neural network technology for image compression," Bolton Institute, UK. This paper appears \in broadcasting convection, pp.250-257, 1995.

[19] Michael T. Kurdziel. "Image compression and transmission for HF radio system". Harrish corporation RF communication division Rochester, NY. This paper appears in MILLCON, vol 2 on pp.1281-1285, 2002.

[20] Aaron T. Deever and Sheila S. "Lossless image compression with projection based and adaptive reversible integer wavelet transform". This paper appears in IEEE transactions of image processing, vol 12, 2003.

[21] Jian Li, Caixin Sun. "Partial discharge image recognition influced by fractal image compression ," Department of High Voltage and Insulation Technology, College of Electrical Engineering, Chongqing University. China. This paper appears in Dielectric and electrical insulation, IEEE transactions, vol 15, pp. 496-504, 2008.

[22] Guan-Nan Hu, Chen-Chung Liu, Kai-Wen Chuang, Shyr-Shen Yu, Ta-Shan Tsui, "General Regression Neural Network utilized for color transformation between images on RGB color space," Proceedings of international conference on machine learning and cybernatics, pp. 1793 – 1799, 2011.

