Satellite Image Encryption Using Neural Networks Backpropagation

I. A. Ismail[1], Galal H. Galal-Edeen[2], Sherif Khattab[2], and Mohamed Abd Elhamid M. El Bahtity[2]
[1] Emeritus dean of faculty of computer and information, Zagazic University, Egypt, amr44_2@hotmail.com
[2] Faculty of computers and information, Cairo University, Egypt
E-mail: {, galal@acm.org, s.khattab@fci-cu.edu.eg, mohbahtity@gmail.com}

Abstract: The main goal of this paper is to investigate the applicability of a back-propagation artificial neural network on the encryption of huge-sized satellite images. The central contribution is using fixed, arbitrary keys in the training process as in classical symmetric and asymmetric cryptography. The used network is of *N*x*M*x*N* neurons representing the input, hidden, and output layers, respectively. The network is trained by adjusting the weights while the bias is given a constant value between 0 and 1 after normalizing the values. A bias is determined. The bias between the input layer and the hidden layer works as the first key *(K_1)*, while the bias between the hidden layer and the output layer represents a second key *(K_2)*.

The training method uses *K_1*, K2, or both and is done using images of small sizes to improve speed. Then, the network is used to encrypt and decrypt normal satellite images. Numerous trials were done for different satellite optical and SAR images and the goodness of fit (quality of decryption) between the original images and the decrypted ones was at least 98%, even for the images that the network was not previously trained to decrypt. It was also found that the network is not affected by geometrical image distortions like translation, size, and rotation.

Keywords: Cryptography, image encryption, neural networks, image processing

I. INTRODUCTION

Cryptography is the study of mathematical techniques related to aspects of information security, such as confidentiality, data integrity, and entity authentication. The first try for using neural cryptography started in January 2002 by the Physicists Kanter and Kinzel. They introduced a new key exchange protocol between two parties A and B. Their method was based on the outcome that two neural networks are able to synchronize to by mutual learning [1].

Synchronized firing (SF) has been observed among cultured cortical neurons, and it is believed that it serves a prominent role in information processing functions of both sensory and motor systems [2]. Synchronization of neural networks applied to cryptography and used for creation of a secure cryptographic secret-key using a public channel [3-6].

In artificial neural networks (ANN), feed-forward Multilayer perceptions (MLPs) utilize the back-propagation (BP) training algorithm [7]. Backpropagation is one of the most commonly used supervised ANN models. A back-propagation network uses the back-propagation learning algorithm to learn a key to be used for encryption and decryption [8].

In this paper, a simple Multi Layer Perception (MLP) network is used for satellite image encryption. This network consists of *N* elements as an input layer that feeds a hidden layer of *M* neurons, which then feed an output layer that has the same number of neurons as the input layer. If the input image is for example 50x50 pixels, it will be segmented into *L* sub-images, each of size 5x5 elements (*N=25*), for instance. Each sub-image is fed as an input to the network**.** The input and hidden layers represent the sender (encryption part), and the receiver consists of the hidden and output layers.

The back-propagation algorithm is used for adjustment of the weight coefficients of the neuron network. The MLP network was tested using different images, some of which were video images and others were satellite images.

The rest of the paper proceeds as follows. The following section contains the structure of the MLP network. Experiments and results are included in Section *III*. Finally, section *IV* contains conclusions and future work.

II. IMAGE ENCRYPTION USING FIXED KEYS

This paper introduces a new idea for using neural back-propagation (NBP) in cryptography; we can securely exchange data between two sites *S* and *R* using NBP. The site S represents the sender (encryption), and the site R represents receiver (decryption). Weight and bias of the NBP represent public and private keys, respectively. The public key is given, the net trained after the weight initialized and the bias may be taken a constant value after normalizing the values.

The bias between the input layer and the hidden layer is called bias1 (K_1), and the bias that lies between the hidden layer and the output layer is called bias2 (K_2). The encryption key is at least one bias (bias1 or bias2 or the two biases). The keys K_1 or K_2 or both will be constant (not updated) during training. The number of keys is determined according to the number of hidden layers in the network configuration. The key length depends on the number of neurons in the hidden layer for bias1 and the number of neurons in the ouput layer or input layer, for bias2. The keys must be numeric; if the keys are characters or strings, straightforward ASCII subsitiuion is applied. The keys are driven from training data.

The main contribution of this paper is to introduce and evaluate a new technique for employing NBP for symmteric cryptography with fixed, arbitrary keys, whereby the sender (*S*) and the receiver (*R*) agree on the keys to be used in NBP training. Our technique can also employ NBP to act as asymmteric cryptography, whereby the weights represent the public key, and the bias represents the private key.

This work assumes a network configuration of *NxMxN* neurons representing the input, hidden, and output layers, respectively. The back-propagation algorithm is exploited for training purpose. The sigmoid function is used for hidden layer and a linear function for the output layer. The sigmoid function is a differentiable function that maps neuron input from the interval $(-\infty,\infty)$ to a new interval $(0,1)$. The output of each hidden neuron is defined using the squashing function:

$$Z=1/(1+\exp[-(x_1v_1+x_2v_2+\ldots+x_nv_n+b_1)]) \quad (1)$$

The linear output function produces values inside the range (–1: +1). This linear activation function increases the dynamic output range and speeds up the convergence rate. The output of each neuron in layer 2 is simply found by:

$$Y=m(z_1w_1+z_2w_2+\ldots+z_Mw_M+b_2)+c \quad (2)$$

where *m* is the slope of the straight line, and C represents the cut from the *y* axis. If the minimum and maximum values of y are *a1* and *a2,* respectively, then the value of *C* was found to be *(a1+ a2)/ (a1- a2)*.

Figure 1 shows the transfer of the input image through the MLP network (BP). Equations 1 and 2 are replaced with Equations 3 and 4, respectively, whereby Equation 3 represents ciphering and 4 represents deciphering.

$$Z(encipher)=1/(1+\exp[-(x_1v_1+x_2v_2+---+x_nv_n+K_1)]) \quad (3)$$

$$Y(decipher)=m(z_1w_1+z_2w_2+---+z_Mw_M+K_2)+c \quad (4)$$

In what follows, we describe the proposed training, encryption, and decryption algorithms.

Training algorithm. The training algorithm is based on the adaptive momentum training algorithm [9] and is defined as follows.

(1) The keys are normalized to obtain real values within the range [0, 1] with the equation:

$$\text{Norm}(K)=k_i/\max(k_i),\ 1<=i<=N \text{ or } M \quad (5)$$

Where *N* is the number of input/output layer neurons, and *M* the number of hidden layer neurons.

(2) Initialize the weight matrix V and W randomly with values ranged from 0 to 1.

(3) Segment the input image into L sub-images, each has N picture elements (pixels).

(4) The input vector $X_k=(x_1,x_2,\ldots,x_N)$ is computed from each sub-image and fed to the network as the input layer. The vector X_k is computed by normalizing the sub-image by dividing each pixel's value by the maximum possible pixel value (e.g., 255) or using a simple relation:

$$X_k(i)=1/[1+\exp(-[x_i-\mu_k]/2\sigma_k)] \quad (6)$$

Where μ_k and σ_k are the mean and the standard deviation of X_k , respectively, and k is the index of the input sub-image.

Then, the output of hidden layer $(z_1,z_2,\ldots,z_M)$ is computed after being activated by Equation (1).

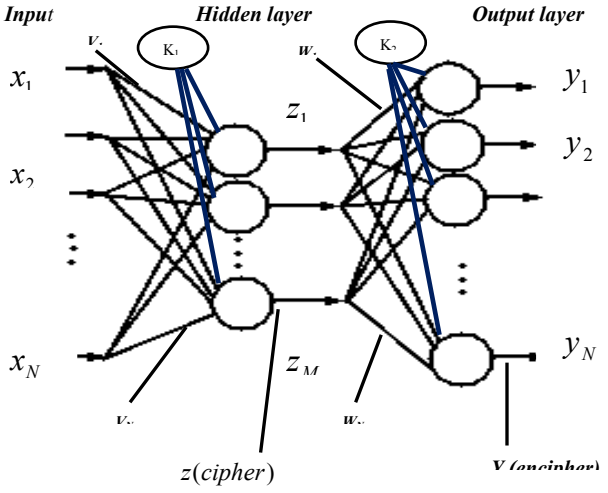

Fig. 1 The structure of the neural network used in image encryption and decryption.

(5) The response of output layer $(y_1,y_2,\ldots,y_N)$ is computed in terms of $(z_1,z_2,\ldots,z_M)$ and the weighting matrix *W* after being activated by a linear function using Equation 2 .

(6) The difference between the desired output (original sub-image) and the actual output $(y_1,y_2,\ldots,y_N)$ in step (n) is calculated using this relation,

$$\Delta(n) = [\sum_{i=1}^{N} (x_i - y_i)^2\ ]^{1/2} \quad (7)$$

(7) Since the output coefficient $w_{p,q}(n)$ represents the weight from a hidden neuron "*p*" to an output neuron "*q*" before the weight adjustment, then $w_{p,q}(n+1)$ after weight adjustment can be computed as follow,

I- The error δ_q signals for neuron "*q"* in output layer is computed as,

$$\delta_q = m(x_q - y_q) \quad (8)$$

II- The updated weight $w_{p,q}(n+1)$ at step $n+1$ is computed as:

$$w_{p,q}(n+1) = w_{p,q}(n) + \Delta w_{p,q}(n+1) \quad (9\text{-a})$$

$$\Delta w_{p,q}(n+1) = \eta \delta_q z_p + \alpha\ [\Delta w_{p,q}(n)], \quad (9\text{-b})$$

where η is the learning coefficient (typically 0.01 to 1.0), and α the momentum coefficient (usually set to around 0.9).

(8) The weight $v_{i,p}(n)$ between hidden neuron p and the i^{th} input node is then adjusted similar to step (7) with changing z_p to x_i, and δ_q to δ_p. However, the error signal δ_p for hidden neuron p is calculated using the following relation:

$$\delta_p = z_p(1 - z_p)[\sum_{k=1}^{N} \delta_k w_{p,k}] \quad (10)$$

(9) Let $\Delta(n)$ and $\Delta(n+1)$ denote the old and new errors respectively. If $\Delta(n+1) > 1.04[\Delta(n)]$, the new weights, keys (constant), output, and error are kept as old values, and α is changed to 0.7α. If $\Delta(n+1) <= \Delta(n)$, the new weights, keys (constant), output, and error are updated to the new values, and α is changed to 1.05α.

(10) The above four steps (step 6-9) are repeated until either the actual error $\Delta(n)$ is less than a predefined value or the number of iteration (epoch) is equal to some value.

The encryption algorithm works as follows.

(1) The input image is segmented into L windows, or sub-images, each of size $L_1 x L_2 = N$ pixels. Then, each sub-image is normalized, because neural networks work better if the inputs and outputs lie between 0 and 1 [11].

(2) The N normalized sub-images are input to the network, and the output of the hidden layer is computed using Equations 3 and 4. The size of the input image is NL, hence, the output of the hidden layer is a vector of size ML, where M is the number of hidden neurons.

(3) The N inputs are 8-bit integer numbers, and the outputs of hidden layer are M real values to transmit as the cipher text. In order to obtain true image encryption, the outputs of the hidden layer are quantized before transmission. This operation is done by rounding after multiplying each neuron's output by 255.

(4) The encrypted image is obtained by transforming the outputs of the hidden layer from a one-dimensional array into a two-dimensional matrix.

The decryption algorithm works as follows.

(1) The M elements of the encrypted image are input to the output layer of the decryption unit, and the response of the output layer $(y_1, y_2, ..., y_N)$ is computed using Equation 1.

Using the response of the output layer, which is of size NL pixels, the decrypted image is extracted by transforming the array into a two-dimensional matrix.

The structure of the network $NxMxN$ has been used with different numbers of hidden neurons $M = 4, 6, 9, 16\ and\ 25$ to select the best structure for image encryption purpose; there is no certain method or approach to determine the best structure [10].

Examination of the relationship between the network structure and the accuracy of the encryption is reported in the next section using MATLAB experiments. It is important to mention here that the size of the input vector was fixed to 25 elements, and the network was trained only to 75 images, all of size 50x50 pixels with 256 gray levels.

III. EXPERIMENTS AND RESULTS

Six sources of satellite images were used in training and testing the proposed encryption/decryption algorithms. All images have 256 gray-levels. **Source1** images were captured at a photographic shop for different human faces (women, men, and children) and were digitized to 50x50 pixels with 256 gray levels. **Source2** images are 48 images of 6 aircraft models; each has 8 snapshots in different orientations, and all images are of size 50x50 pixels. **Source3** images were collected from public magazines, stories, and cartoons, and were all digitized to 512x512 and 50x50 pixels. **Source4** images were downloaded from the Internet[1]. **Source5** images were electro-optical satellite images from the the Ikonos

[1] From http://www.fotosearch.com/photos-images/grayscale.html.

satellite[2] with size 512x512 pixels. **Source6** images are Synthetic Aperture Radar (SAR) images[3].

The images from the 6 sources mentioned above were distributed over 6 sets. **Set1** contained 108 images from **Source1**, **Source2**, and **Source3. Set2** consisted of 60 images from **Source4** and different figures of faces (women, men, and children) from **Source1**. **Set3** contained 4 images of size 512x512 from **Source3**. **Set4** contained 14 Synthetic Aperture Radar (SAR) images from **Source6**. **Set5** consisted of 12 images of satellite electro-optical images from **Source5**. **Set6** was used for training and contained 50x50 images from **Source1**, **Source2**, and **Source3**.

The experiments were done on a PC with the following configuration of H/W and S/W: Intel(R) core(TM) i3 CPU M330 @ 2.13GHz, 4.00 GB RAM, and 64-bit OS. The algorithms were implemented in Matlab 6.0.0.88 Release 12.

To measure the performance of the encryption algorithm, the goodness of fit ϕ_k between the original and the decrypted images are calculated using the simple relation:

$$\phi_k = 1 - (\| X_l - Y_l \| / \| X_l \|), \quad (11)$$

$$\| X_l \| = (\sum_{i=1}^{NL} x_i^2)^{1/2} \text{ , and } \| Y_l \| = (\sum_{i=1}^{NL} y_i^2)^{1/2}$$

where X_l represents the input sub-images of size L, and Y_l is the corresponding decrypted image.

Different numbers of hidden neurons (4, 6, and 25) with fixed input and output nodes were used to study the effect of the number of hidden neurons. It was found that as the number of the hidden neurons increased, the goodness of fit increased, as listed in Table 1.

Table 1. Effect of the size of the hidden layer. The structure of the network is 25xHx25 with varying size of hidden layer (H) (4, 6, and 25), and the number of iteration in the training algorithm =30.

Number of neurons in hidden layer	Number Of Images (trained and new) in the three categories of goodness of fit (GOF%):						number of images	Time Train (Minute)
	82-90		91-96		97-100			
	Train	New	Train	New	Train	New		
4	75	108	0	0	0	0	183	54
6	44	61	31	47	0	0	183	60
25	0	0	0	0	75	108	183	70

It was found also that the network (25x25x25) was suitable for a bigger sized satellite image with keys, K_1, K_2 or both K_1 and K_2.

The satellite images in Set4 and Set5 were tested using the neural BP network with fixed keys, K1, K2, or both. The experiments resulted in a minimum of 98% of goodness of fit. The quality of the decrypted images for Set4 when training was done with fixed K1 were 99, 99, 98, 99, 98 ,99, 99, 99, 98,98, 99, 99, 99 and 99, respectively, as shown in Table 2, Figure 2, and Figure 3.

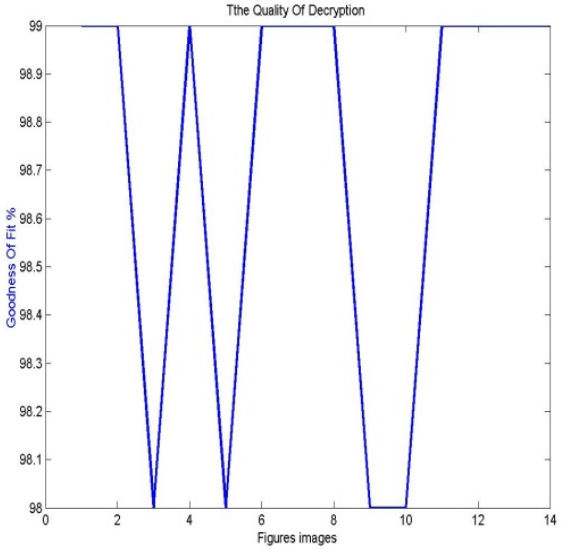

Figure 2 Quality of decryption (set4) for K1

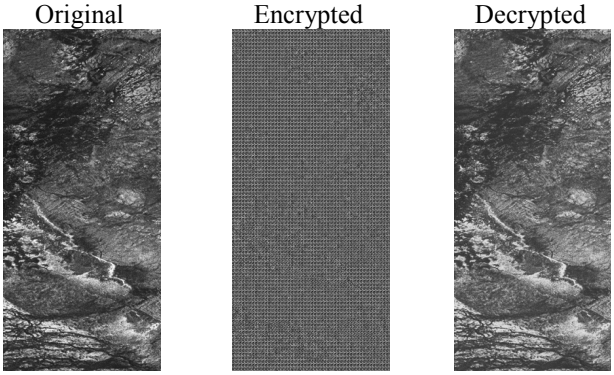

Figure 3 Samples of images (South-Central Egypt) of Set4

The quality of decrypted images of **Set5** when training was done with fixed K_1 were 99, 99, 99, 99, 99, 99, 99, 99, 99, 98, 99, and 99, as shown in Table 3, Figure 4 and Figure 5.

Table 2. Effect of key for Set4 images (14 images).

Keys used	Quality of decryption (GOF) % for Set4					Total number of images
	96	97	98	99	100	
K1	0	0	4	10	0	14
K2	0	0	7	7	0	14
K1K2	1	1	6	6	0	14

[2] From http://www.mapmart.com/Products/SatelliteImagery/IKONOS.aspx
[3] From http://www.jpl.nasa.gov/radar/sircxsar/

Table 3. Effect of key for Set5 images (12 images).

Keys used	Quality of decryption (GOF) % for Set4					Total number of images
	96	97	98	99	100	
K1	0	0	1	11	0	12
K2	0	0	1	11	0	12
K1K2	0	0	2	10	0	12

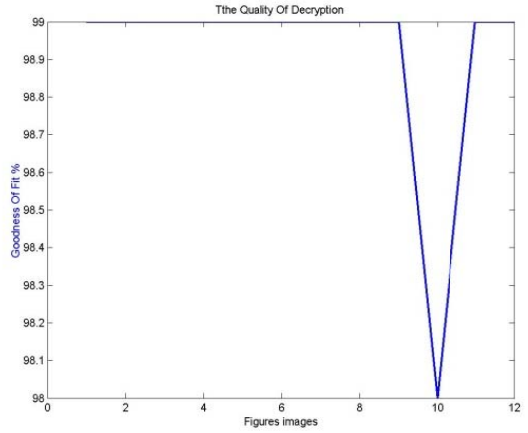

Figure 4 Quality of decryption (set5) for K1

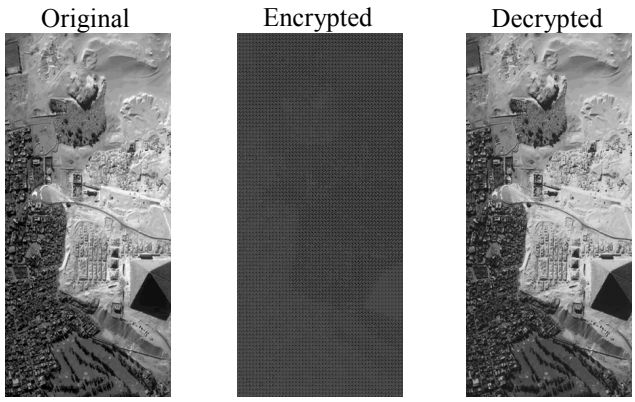

Figure 5 Samples of images (Pyramids Egypt) of Set5

The reconstructed satellite images of **Set4** and **Set5** had goodness better than images belonging to sets 1, 2, and 3.

VI. *CONCLUSION AND FUTURE WORK*

A feed–forward Multilayer perception (MLP) that utilizes back-propagation (BP) was found to be suitable for image encryption applications using fixed, arbitrary keys as in classical cryptography. The length of keys depends on the structure of the neural network. The proposed network and algorithms worked well on images even the ones that the network was not trained on. The presented neural network was not affected by the geometrical distortions of the input image, such as translation, scale, and rotation. The presented network proved to be suitable for encryption of giant-sized pictures, such as satellite images, with quality of decryption ratio of at least 98%. The reconstructed satellite images had goodness of fit better than that of images having small size, and each took about 1.6 minutes for reconstruction. It is important to mention here that the network was not trained on satellite images.

For future work, the efficiency and performance of the proposed algorithms will be evaluated using statistical measures of randomness, and the proposed algorithms will be compared against traditional image encryption algorithms.

REFERENCES

[1] N. Prabakaran, P.Loganathan and P.Vivekanadan, "Neural Cryptography with Multiple Transfers Function and Multiple Learning Rule", International Journal of Soft Computing 3(3):177-181, 2008, ©Medwell Journals, 2008

[2] Erol Gelenbe, Stelios Timotheou, "Random Neural Networks with Synchronized Interactions", *Neural Computation* 20, 2308–2324 (2008), © 2008 Massachusetts Institute of Technology

[3] I. Kanter, W.Kinzel, "The Theory of Neural Networks and Cryptography", Quantum Computers and Computing, V. 5, No.1, 2005

[4] R. M.Jogdand1 and Sahana S.Bisalapur, "DESIGN OF AN EFFICIENT NEURAL KEY GENERATION", International Journal of Artificial Intelligence & Applications (IJAIA), Vol.2, No.1, January 2011

[5] Roland E.Suri, Terrence J. Sejnowski, "Spike propagation synchronized by temporally asymmetric Hebbian learning", Biol. Chern. 87, 440–445 (2002) DOI 10.1007/s00422-002-0355-9, © Springer-Verlag 2002

[6] David Norton and Dan Ventura, "Preparing More Effective Liquid State Machines Using Hebbian Learning", 2006 International Joint Conference on Neural Networks Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada July 16-21, 2006

[7] Joarder Kamruzzaman Monash, "Artificial neural networks in finance and manufacturing", Idea group publishing, 2006, ISBN 1-59140-672-2

[8] Khalil Shihab, "A Cryptographic Scheme Based on Neural Networks", Proceedings of the 10th WSEAS International Conference on COMMUNICATIONS, Vouliagmeni, Athens, Greece, July 10-12, 2006 (pp7-12)

[9] Enrique Castillo, Bertha Guijarro-Berdinas, Oscar Fontana-Romero and Amparo Alonso-Betanzos, "A Very Fast Learning Method for Neural Networks Based on Sensitivity Analysis", Journal of Machine Learning Research 7 (2006) 1159–1182

[10] Taskin Kavzoglu, "Determining Optimum Structure for Artificial Neural Networks", In Proceedings of the 25 Annual Technical Conferences and Exhibition of the Remote Sensing Society, Cardiff, UK, pp. 675-682, 8-10 September 1999.

[11] S. Anna Durai, and E. "Anna Saro, "Image Compression with Back Propagation Neural Network using Cumulative Distribution Function", World Academy of Science, Engineering and Technology 17 2006, (pp60-64)