Steganography Detection using Functional Link Artificial Neural Networks

Ch. Demudu Naidu
Sr. Asst. Professor
Dept of I.T, ANITS,

S. Pallam Setty
Department Of CS & SE
Andhra University

M. James Stephen
Assoc. Prof.
Dept. of I.T, ANITS,

S.K.Prashanth
Assoc. Prof., CSE
Vardhaman

Ch. Suresh
Professor
Dept. of I.T
ANITS

ABSTRACT
Security in message transfer over the netwoek has been a consistent challenge in the field of I.T. Cryptography is one of very much spoken solution. Security of messages that are being transferred is very important and experts have lot of work to think of new techniques and approaches in cryptography. At the same time cryptanalysts also have very important job to detect and reveal and then decode the coded messages. Steganography is another additional method for better secure up messages which goes hand by hand with cryptography, and reveal of such a message is not easy. In this paper, we presented a new approach known as Steganography detection using Functional Link Artificial Neural Networks that deals with neural network models that are able to detect Steganography content coded by a program Outguess. Neural network methods are flexible in learning various typical problems. In this paper 'Functional Link Artificial Neural Network' is used which is one of the methods for training a neural network. Results in this project show that used model had almost 100% success in revealing Steganography by means of Outguess.

General Terms
JPEG Snoop, Legendre polynomials, Chebyshev polynomials, Power series, Ubuntu.

Keywords
Cryptography, Steganography, Neural Network, Outguess, Functional Link, Artificial Neural Network

1. INTRODUCTION
The purpose of Steganography[1] is to hide the very presence of communication by embedding messages into innocuous looking cover objects, such as digital images. The secret message is embedded in the original cover image by making slight modifications to it. As a result, the stego image is obtained. If you see such a picture, normally you do not recognize that there is a secret message. And this is the point Hacker will go through and will not give the attention to the message. Therefore it is necessary to have a method for its detection. It is important to know the fact that when a picture is coded using OutGuess, no change can be observed between the uncoded and coded image to the naked eye. Because there is no change in the frequency of the image. So, a procedure should be followed to detect such an image. When an image is coded, the hidden data is actually present in the least significant bits of the image. By this we can say that if we calculate the Huffman coding of the image there is a change in the pattern of the AC and DC bits for a coded and uncoded image. Using this concept, we feed the neural network with the Huffman code of the uncoded images i.e, the data set to be given to train the neural network is made of Huffman codes. After training the neural network with uncoded images, it identifies a coded image by showing a variation in its output. A neural network gives an output close to zero (0) if the image is uncoded. If the image is coded, the output is close to one (1)

2. EXISTING SYSTEM
Outguess preserves statistics based on frequency counts. As a result statistically tests based on frequency counts are unable to detect the presence of steganographic content. Results are unable to reveal steganography content because Out Guess finds out maximal length of the message before inserting into image. There is lot of literature available in this field, some of them are 'Detection of Steganography Inserted by OutGuess and Steghide by Means of Neural Networks' by Oplatkova, Z. Holoska, J.; Zelinka, I.; Senkerik, R.[10], 'Using an Aritificial Neural Network to Detect the Presence of Image Steganography' by Chandrababu, Aron [11], New Steganalysis Method using GLCM and Neural Network by Sedighe Ghanbari, Manije Keshtegary, Najme Ghanbari [12]. But there is no such software which detects Steganography done by Out Guess at present.

3. PROPOSED SYSTEM
We use Functional Link Artificial Neural Network to detect the coded images. The FLANN architecture uses a single layer feed forward neural network and to overcome the linear mapping, functionally expands the input vector. We obtain data for training sets of neural networks using JPEG snoop. JPEG snoop is used to create tables of Huffman's coding. Each column from the table was taken and given to a row one after one. With the help of Huffman's coding. We generate the data sets. These data sets are fed to the neural network by training these data sets to the neural network we are able to say whether the image is clear or coded. There is also a Method for Automatic Identification of Signatures of Steganography Software [9] but our proposed method works as good as any other automatic tools available in the literature.

4. OUTGUESS
Niels Provos created OutGuess [7], which used selective pseudo-random number generator seeding to deterministically offset statistical aberrations caused by steganographic embedding in JPEG images, and also created StegDetect [8], an application which detects various steganography schemes in JPEG with a high degree of reliability.
Out Guess is a universal steganographic tool that allows the insertion of hidden information into the redundant bits of data sources. The nature of the data source is irrelevant to the core of Out Guess. The program relies on data specific handlers that will extract redundant bits and write them back after modification. In this version the PNM and JPEG image

formats are supported[3]. The JPEG format is currently the most common format for storing image data. It is also supported by virtually all software applications that allow viewing and working with digital images.

Recently, several steganographic techniques for data hiding in Out Guess [4]. In all programs, message bits are embedded by manipulating the quantized DCT coefficients. Out Guess embed message bits into the LSBs of quantized DCT coefficients. The program relies on data specific handlers that will extract redundant bits and write them back after modification. It is also an open source tool and is obtained as a direct download from the official website of outguess. It is UNIX compatibility software. So it should be installed prior to its operation. The following are the sequence of operations to install Outguess. Open Terminal Window Login as Super User using the bash command 'su' and then enter the 'password'. Change the present working directory to the location where the outguess tar.gz file is located. Now extract the file using the command 'tar –xvzf *filename*'. Now change the present working directory to the extracted file. Now type './configure'. Now 'make' and then 'make install'. In some versions of Linux these set of operations may cause exceptions. Then the following command can be used after setting the present working directory to the file location To embed a file into the image the following command is used out guess -k "my secret key" -d hidden.txt demo.jpg out.jpg Here "my secret key" represents the key with which the file can be extracted later from the image later and it takes the role of security handler. demo.jpg represents the image in which the text is to be hidden. hidden.txt represents the data in .txt format which is to be hidden into the image. out.jpg is the intended name of the image file obtained after the data is hidden into the image. To extract the hidden data the following command is used.outguess -k "my secret key" –r out.jpg hidden.txt "my secret key" represents the key with which the file is steganised. out.jpg is the image file obtained as output after hiding the data into the image.

5. UBUNTU

Ubuntu is a computer operating system based on the Debian Linux distribution and distributed as free and open source software. The original aim of the Ubuntu team was to create an easy-to-use. Outguess can run in this linux platform.Users can download a disk image (.iso) of the CD, which can then either be written to a physical medium (CD or DVD), or optionally run directly from a hard drive

6. JPEG SNOOP

A program JPEG snoop was used which is able to work with extended information in image, video and text files. JPEG snoop is able to extract such a information like:

Matrices of quantization tables of colourness and brightness Information about reduction of colour parts:

Quality of JPEG image.
EXIF information.
RGB histograms.
Tables of Huffman's code.

JPEG Snoop is very simple and user-friendly device to test and use. The image is opened into the JPEG Snoop window and automatically the different hidden encoding for the image are displayed as results and also it provides an option for saving the results in a .txt format which enables the user to work over the obtained results.

7. HUFFMAN'S CODING

Huffman coding are pervasive throughout computer science. This coding scheme is not limited to encoding messages in English text. In fact, Huffman coding can be used to compress parts of both digital photographs and other files such as digital sound files (MP3) and ZIP files. If you have ever used a JPEG picture file then you have probably used Huffman coding[6] without even knowing it.

In the case of JPEG files the main compression scheme uses a discrete cosine transform, but Huffman coding is used as a minor player in the overall JPEG format. There are of course many other file compression techniques besides Huffman coding, but next to run-length encoding, Huffman coding is one of the simplest forms of file compression. If you want to understand other more sophisticated compression schemes such as arithmetic coding and Lempel-Ziv-Welch coding you will be better served by first mastering Huffman coding. Huffman coding can be used effectively anywhere there is a need for a compact code to represent a long series of a relatively small number of distinct bytes

Table 1. Table of Huffman's coding – clear picture

	DC, Class 0	**DC, Class 1**	**AC, Class 0**	**AC, Class 1**
1 Bit	0	0	0	0
2 Bit	12	59	46	49
3 Bit	73	16	12	17
4 Bit	12	14	24	18
5 Bit	3	9	9	10
6 Bit	0	2	3	4
7 Bit	0	0	3	0
8 Bit	0	0	1	1
9 Bit	0	0	1	0
10 Bit	0	0	0	0
11 Bit	0	0	0	0
12 Bit	0	0	0	0
13 Bit	0	0	0	0
14 Bit	0	0	0	0
15 Bit	0	0	0	0
16 Bit	0	0	0	0

Table 2: Table of Huffman's coding – coded picture

	DC, Class 0	DC, Class 1	AC, Class 0	AC, Class 1
1 Bit	0	0	0	0
2 Bit	15	80	42	63
3 Bit	73	11	10	11
4 Bit	7	7	25	16
5 Bit	4	1	11	6
6 Bit	0	0	4	3
7 Bit	0	0	4	0
8 Bit	0	0	1	1
9 Bit	0	0	1	0
10 Bit	0	0	0	0
11 Bit	0	0	0	0
12 Bit	0	0	0	0
13 Bit	0	0	0	0
14 Bit	0	0	0	0
15 Bit	0	0	0	0
16 Bit	0	0	0	0

8. TRAINING SETS

To generate the training sets for training the neural networks, different series of photographs[5] were considered. On a whole 200 images were considered with varying resolutions. The resolutions taken to accomplish the results were

S 1024X768
S 1280X1024
S 2048X1536
S 2592X1944
S 3872X2592 [1]

These five resolutions are applied to all the images using photo editing tools and the corresponding Huffman coding are extracted using the JPEG Snoop tool for both the clear image as well as the coded image which have been coded through the Outguess tool. The message resembles to be unique in each image due to pseudo random generator of numbers.

After the JPEG Snoop created the tables for the coded image i.e. Huffman's Coding, each column from the tables was taken and given to a row one after one. This led to a vector of length equal 40. Examples of clear and coded inputs in a training set are in Fig. 3 and Fig. 4. It is a matrix of 5 individual inputs of length 40. These inputs are then given on the input layer of neural network. Output was 0 in the case of clear input and nearly or equal to1 in the case of coded input

Table 3: training set for clear image

Image (1024X768)€
{0,12,73,12,3,0,0,0,0,0,0,59,16,14,9,2,0,0,0,0,0,46,12,
24,9,3,3,1,1,0,0,49,17,18,10,4,0,1,0,0}

Image (1280X1024)€
{0,11,75,11,2,0,0,0,0,0,0,61,16,13,8,2,0,0,0,0,0,45,13,
24,9,3,3,1,1,0,0,51,17,17,9,4,0,2,0,0}

Image (2048X1536)€
{0,12,80,7,1,0,0,0,0,0,0,70,16,10,4,1,0,0,0,0,0,44,11,
33,8,2,1,1,0,0,0,66,13,14,4,2,0,1,0,0}

Image (2592X1944)€
{0,11,78,8,2,0,0,0,0,0,0,66,17,11,5,1,0,0,0,0,0,44,12,
30,9,2,2,1,1,0,0,61,14,15,6,3,0,1,0,0}

Image (3872X2592)€
{0,12,76,9,3,0,0,0,0,0,0,64,17,12,6,1,0,0,0,0,0,45,12,
27,9,3,2,1,1,0,0,58,15,16,7,3,0,1,0}

Table 4: training set for coded image

Image (1024X768)€
{0,15,73,7,4,0,0,0,0,0,0,80,11,7,1,0,0,0,0,0,0,42,10,25,11,4,4,1,
1,0,0,63,11,16,6,3,0,1,0,0}

Image (1280X1024)€
{0,16,73,7,4,0,0,0,0,0,0,81,11,6,1,0,0,0,0,0,0,43,
10,26,11,3,3,1,1,0,0,67,11,14,5,3,0,1,0,0}

Image (2048X1536)€
{0,16,75,5,3,1,0,0,0,0,0,88,8,3,0,0,0,0,0,0,0,41,8,38,8,2,2,1,0
,0,0,78,7,10,3,2,0,0,0,0}

Image (2592X1944)€
{0,16,74,6,4,1,0,0,0,0,0,86,9,4,1,0,0,0,0,0,0,41,8,34,10,2,3,1,
0,0,0,75,8,11,4,2,0,0,0,0}

Image (3872X2592)€
{0,15,74,6,4,1,0,0,0,0,0,84,9,5,1,0,0,0,0,0,0,42,9,31,10,3,3,1,
1,0,0,71,9,12,5,2,0,1,0,0}

Once the training sets are obtained we need to feed it to the neural networks to obtain the weights of the connections between the neurons. This can be done by using Back propagation algorithm.

Fig.1-Clear Image

Fig.2-Coded Image

9. FUNCTIONAL LINK ARTIFICIAL NEURAL NETWORK

Functional Link Artificial Neural Networks are a popular type of network that can be trained to recognize different patterns including images, signals, and text. FLANN architecture for Steganography detection coded in the images is the single-layer feed forward neural network consisting of one input and an output layer. The FLANN generates output (0 or 1) by expanding the initial inputs (bits in datasets) and then processing to the final output layer. Each input neuron corresponds to a component of an input vector. The output layer consists of one output neuron that computes the output (0 or 1) as a linear weighted sum of the outputs of the input Layer. The genrall architecture of Functional Link Artificial Neural Network (FLANN) is shown in Fig.3

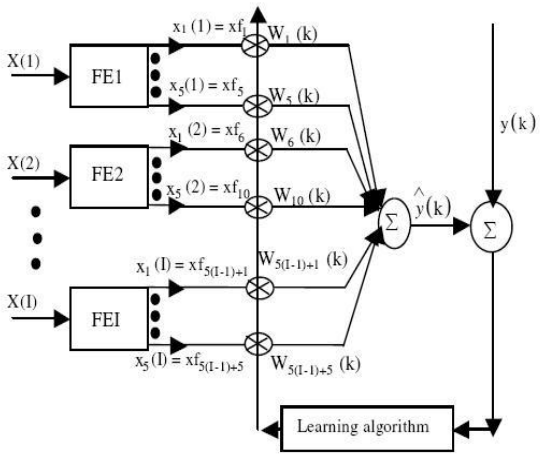

Fig.3 FLANN Architecture

The functional expansion takes bits of input data(200 bits) for each image and expands the input vector to 800(200*4)values. Initially weights are generated randomly for each of the above outputted values. Then we apply logarithms (the reason for applying logarithms is that when we raise the outputted values to some power the value becomes very large and hence in order to minimize the values and fit them within the range we apply logarithms) for the outputted values and then multiply them with the random weights.

Now each of the computed values are added together and a sum is generated. This generated sum becomes the output. We find the difference between the obtained output and expected output and this difference is the error. Now we check whether the error is within the specified limit, if it is not within the limit then we subject it to learning algorithm. If it is within the range then we input the next image and repeat the above process. Now we multiply the error with a random weight and compute a very small amount of weight dw. Then depending on the error we add or subtract this small weight dw to the random weight.

Now we calculate the output and the error for these modified weights and repeat checking the error and deciding whether to apply the learning algorithm or processing the next image. If the checking of errors for all the images is done then the training completes. We take some images which are not part of the training process and give them as input and compute the output. If the computed output is close to 0 then the image is said to be clear. If it is nearer to 1 the image is said to be coded.

There are three different functional expansion of the input pattern in the FLANN. They are Chebyshev, Legendre and Power Series, corresponding networks named as C- FLANN, L-FLANN and P-FLANN respectively.

9.1 Chebyshev polynomials [5]

$$T_0(x) = 1, T_1(x) = x, T_2(x) = 2x^2 - 1,$$
$$T_3(x) = 4x^3 - 3x, T_4(x) = 8x^3 - 8x^2 + 1.$$

Higher order Chebyshev polynomials may be generated by the recursive formula given by:

$$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x), n \geq 2, (-1 \leq x \leq 1).$$

9.2 Legendre polynomials [5]

$$L_0(x) = 1, L_1(x) = x, L_2(x) = (3x^2 - 1)/2,$$
$$L_3(x) = (5x^3 - 3x)/2, L_4(x) = (35x^4 - 30x + 3)/8.$$

Higher order *Legendre* polynomials may be generated by the recursive formula given by:

$$L_n(x) = [(2n-1)xL_{n-1}(x) - (n-1)L_{n-2}(x)]/n,$$
$$n \geq 2, (-1 < x < 1).$$

9.3 Power series (*x* =any value) [5]

Convergence of the series will depend upon the values of *x* that we put into the series. A power series may converge for some values of *x* and not for other values of *x*

$$P_n(x) = x^n, n \geq 0.$$

10. EXPERIMENTAL RESULTS

Input: For the experimental purpose, we have taken the input (datasets) from the Table 3 and Table 4 for the images Image1 and Image2.

Output: The experimental results had shown that the proposed system could detect whether the image is clear or by coded by using the obtained output value. If the output is equal or nearer to 1, then it is a coded image. If the output is 0 or nearer to 0, then it is a clear image.

We coded the algorithms in JAVA and developed a standalone system and created a user interface for the ease of use. Results in this project show that used model had almost 100% success in revealing Steganography by means of Outguess

11. CONCLUSION

This project deals with detection of steganography content in images inserted by program Outguess. The revealing of it is done by means of the system developed based on Functional Link Artificial Neural Networks. Results show that used neural networks has a big success, almost 100%.

The further work supposes to learn other types of neural networks also to find their ability of adaptation on such problems. The parallel step is also to try to learn neural networks on other steganographic methods – e.g. inserted by a program Steghide.

12. REFERENCES

[1] Zuzana Oplatkova, Jiri Holoska, Ivan Zelinka, Roman Senkerik, Steganography Detection by Means of Neural Networks 19th International Conference on Database and Expert Systems Application, 2008 September 01- September 05, ISBN: 978-0-7695-3299-8

[2] R.J. Anderson and F.A.P. Petitcolas, "On the Limits of Steganography", IEEE Journal of Selected Areas in Communications, Special Issue on Copyright and Privacy Protection, vol. 16(4), pp. 474−481, 1998.

[3] Software Outguess www.outguess.org

[4] Provos, N. Defending Against Statistical Steganalysis. Proc.10th USENIX Security Symposium. Washington, DC, 2001.

[5] A Novel Neural Network Approach For Software Cost EstimationUsing Functional Link Artificial Neural Network (FLANN)- B.Tirimula Rao, B.Sameet, G.Kiran Swathi.

[6] Huffman tables homepage.smc.edu/kennedy_john/HUFFMAN.PDF

[7] N. Provos. Defending against statistical steganalysis. In Proceedings of the 10th USENX Security Symposium, Washington D.C., August 2001

[8] N. Provos. OutGuess – Steganography Detection. http://www.outguess.org/ detection.php.

[9] Bell, G., Lee, Y.K.. A Method for Automatic Identication of Signatures of Steganography Software. IEEE Transactions on Information Forensics and Security 2010;5(2):354{358.

[10] Oplatkova,Z, Holoska, J. ; Zelinka, I. ; Senkerik, R. 'Detection of Steganography Inserted by OutGuess and Steghide by Means of Neural Networks'Third Asia International Conference on Modelling & Simulation, 2009. AMS '09.

[11] Chandrababu, Aron, Using an Aritiicial Neural Network to Detect the Presence of Image Steganography' A Thesis Presented to The Graduate Faculty of The University of Akron, May, 2009

[12] Sedighe Ghanbari, Manije Keshtegary, Najme Ghanbari, New Steganalysis Method using GLCM and Neural Network, International Journal of Computer Applications © 2012 by IJCA Journal, Volume 42 - Number 7.