Dissertation thesis

ARTIFICIAL INTELIGENCE APPLIED ON CRYPTOANALYSIS AIMED ON REVEALING WEAKNESSES OF MODERN CRYPTOLOGY AND COMPUTER SECURITY

Jiří HOLOŠKA

Supervisor: prof. Ing. Ivan Zelinka, Ph.D.

Consultant: Ing. Zuzana Oplatková, Ph.D.

Tomas Bata University in Zlin
Faculty of Applied Informatics
Department of Informatics and Artificial Intelligence

Zlin, 2011

Acknowledgement:

It is a great pleasure to thank everyone who helped me to write my dissertation successfully. I would like to thank to my supervisor, Prof. Ivan Zelinka, for his excellent guidance and material support.

I would like to express my deepest gratitude to Dr. Zuzana Oplatkova who helped me to overcome practical issue on artifficial neural networks field, patiently corrected my writing and over all helped me with the research far beyond her duty as research consultant. I am obliged to many of my fellow researchers Michal Procházka, Martin Hromada, Roman Šenkeřík, Jozef Fekiač and Michal Pavlech for their supportive comments and good work envinroment they provide during my research experience.

On the very end, I would like to thank my father and mother for all the support they provide me during all my university period.

Theory is when you know something, but it doesn't work. Practice is when something works, but you don't know why. We combine theory and practice: Nothing works and we do not know why. – Unknown

RESUME

Tato disertační práce je zaměřena na odhalování informací ukrytých do multimediálních souborů, především do obrázků. Informace (zprávy) byly ukryty pomocí steganografických metod, které jsou doplňkovou metodou kryptografie. Steganografie společně s kryptografií zvyšují výslednou bezpečnost přenášené zprávy, tj. snižují možnost kompromitace přenášených informací. Staré metody stegoanalýzy jsou postavené na statistickém útoku na změny v koeficientech diskrétní kosinové transformace (DCT). Tyto lineárně klasifikační metody mají ovšem dobře prokazatelný problém při klasifikaci blízko hranice jednotlivých skupin. To má často za následek nepřesné nebo mylné zařazení testovaného vzorku do špatné skupiny (false positive classification). Nejčastěji je to zařazení nosného (nemodifikovaného) souboru do skupiny stego souborů. Hlavním cílem této disertační práce je minimalizovat nebo úplně eliminovat tyto nepřesnosti s využitím umělých neuronových sítí. Tato práce se zabývá detekční metodou založenou na metodách umělých neuronových sítích v několika různých topologiích. Při ověřovacích testech se podařilo dosáhnout téměř stoprocentní úspěšnosti detekce a tím získat slibný základ pro tvorbu samostatné aplikace pro stegoanalýzu.

SUMMARY

The thesis is focused on revealing of hidden information presence in multimedia files, mainly in pictures. This hidden information (messages) was coded in by means of steganography which is an additional method of cryptography. Steganography causes a better security for messages and the detection of such a message is uneasy. Old fashioned detection methods are based on statistical attack upon discrete cosine transformation (DCT) coefficients. These linear clasification methods have been proved to have limitation on clasification close to detection border. This often leads to a false positive clasification result. Mostly it is a clasiffication of cover files into stego files group. Main goal of this research is clasification by means of neural networks aimed to reduce false positive classification results to minimum. Therefore artificial neural network was used in several structures as a detection method in the thesis. Compared to older methods results showed almost 100% success in detection in this case therefore artificial neural network are promising for future design of a universal detector.

CONTENTS:

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

symbol	*unit*	*meaning*
DDSAANN		Deep data structure analysis by artificial neural network
SVM		Support Vector Machine
LSB		Least significant bit
DCT		Discrete cosine transformation
BSD		Berkeley Software Distribution

1 INTRODUCTION

In the current world we cannot imagine our lives without computers. However with the use of computers a question of secure data transfer appears rather soon. Informatinon coding and cryptography is essential, but efficient privacy has been given by encryption and information hiding methods that can be misused for covering criminal activities. Therefore is important to develop tools and methods for forensic analysis.

Steganography [1] and cryptography [2] are normally connected together. Cryptography is effective in the usage of the key and the message is somehow coded. If it is sent unsecurely, an attacker will notice it immedately and will try to decode it. However there is a steganography, which helps with the secure transfer of encoded messages. It codes a message inside of a picture or another multimedia file. If you see a steganographic picture, you will not recognize the secret message inside of picture. And this is the point. Crackers will go through and will not pay attention to the message. Therefore it is necessary to have a method for its detection. To decode a message itself is another challenge, this thesis is aimed to reveal a secret message inside the picture.

To get better understanding of this research, following the paragraphs describe the imaginary case study of the customers data leak. Imagine a company with employees and secret information, there is a customer database located on database server accessible from the employee's terminal. The employee from the customer service department has received an offer from a competitor to steal information about VIP customers from the customer's database. The employee has access to the terminal with the monitored customer services email account (as seen in Fig. 1). The customer service employee saved the internal customer data into a regular email and sent it by the customer department email account to his home computer.

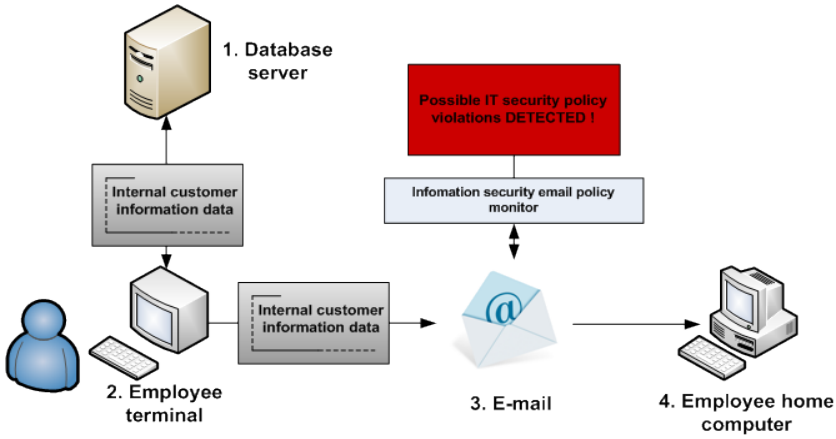


Fig. 1 Message transport trough plain texted email.

However there is an email monitor between the terminal and the email gateway which checks all the outgoing emails for viruses as well as its body plus attachments for internal business information. In this case, the security monitor detected that an email attachment contains sensitive VIP customer information. The security department was immediately informed about this incident and the employee will be charged for information fraud.

In Fig. 2 a similar scenario is shown, but instead of the employee from the first example a more experienced cracker is used. A new customer department employee has intermediate computer skills he is not naive at all. This means that a skilled user is familiar with the computer security policy and the cracker expects some kind of testing of sent messages. Cryptography is strong in the usage of the key and the message is coded. Sending such an unsecure message can cause attention from people who are not supposed to know the secret message. Steganography helps with the secure transfer of secret messages. It codes a message inside the picture, video file or data stream. If you saw picture with steganographic content, you would not recognize that there is a secret message. This is the point.

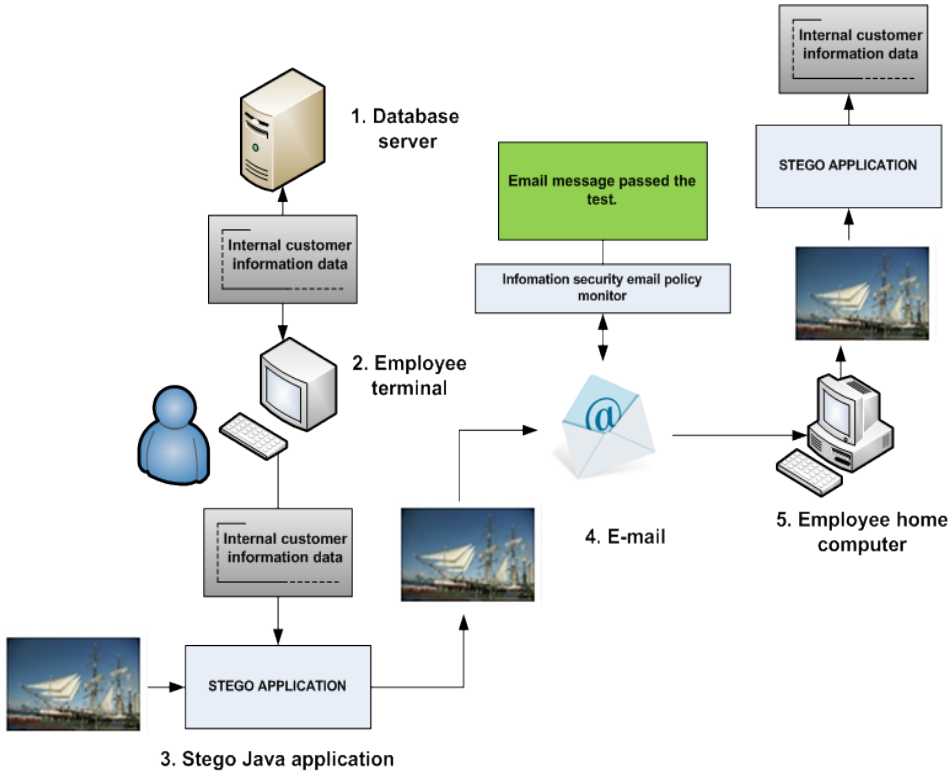


Fig. 2 Message transport with steganography.

The employee will decide to use a Java application downloaded from the Internet after considering the employee's options. The first reason to use a Java application is that he cannot install any application on employee's terminal. Second, that a Java application is multi platform so it is possible to run it on Windows as well as on Unix-like system, which is used in the company.

This whole scenario is very easy. A cracker prepares a few images in JPEG format and a steganographic Java application. Then he embeds a text file containing internal business information into the prepared JPEG images. After that he sends those images to his home email account.

Another scenario follows. An employee registers a new email account with any free email server. Then he writes an email message with the text like *„Hello Bob, there are a few shots from holiday which you asked me for last week. Have a nice day. Mark.“* He attaches images and a steganographic application to this email message. He sends this email to work email. Next day the malicious user will reply to this email with a text like this: *“Hello Mark, I believe that you misspelled the email address because you reached customer services department of AmTrade Inc. Please, check your recipient address and send it again. Have a nice day. P.S. I am sure that Bob will enyoy the pictures.”* Of course, he attaches images with embedded customer information as in previous case. This way the employee marks an email as miss delivered. He avoids attracting attention and sensitive information was delivered into his email account as required.

Not to attract attention is the main goal of steganography. Therefore it is necessary to have a method for its detection, because it is vulnerable to criminals. This thesis deals with such a case – method of detection by means of Artificial Neural Networks (ANN) [3]. It is focused on detection of pictures, which a message by a program OutGuess, Steghide, CipherAWT (F5 algorithm) or PQ algorithm [4] – [17] was coded in.

Firstly, steganography methods and basics of stegoanalysis are mentioned. Secondly, the next part explains the data extraction from the JPEG images and the transformation of information into training sets for neural networks. The description of neural network is described in the next chapter and is followed by the reserach results. The decoding messages, is another challenge, which is not mentioned or examined in this research. The main aim is only to detect cover (clear) or stego (with a message inside) images.

2 STATE OF ART

Records on modern computer based steganography can be found before year 1995, but steganography become more known to public around year 2000. Since then, there has been a constant development. Steganography and steganalysis projects have become very popular as research topics. Among very first released steganalysis tools belonge Stegdetect. when released by Niel Provos. This tool had been developed for next three years along with steganography tool Outguess [7] and became a starting point for many researchers in steganalysis field.

As time goes, technology and steganography become more sophisticated, but in general there are still three main methods of steganoraphy. Injection steganography has been designed to embed secret message payload into cover medium, extend its size, but preserve its functionality when processed by original application. Substitution steganography has more advanced approach to hidden information. It generally rewrites particular bits in original cover medium payload by secret message payload. As well as in injection steganography substitution steganography has to be done in such manner that original application has to be able to process such a modified file without compromising hidden content. Propagation steganography generates its own cover medium instead of using some as an input. The output is usually similar to extravagant drawing or free form of text document carefully balanced to pass over the statistical attacks.

In the same manner, steganalyis has basic techniques to discover the covert communication in transfered data files. The very basic test, which is called visual attack, uses human senses such as sight for descovering irregularities in represented medium, as in this particular case. Such test is limited by human individuality, which means that two people will always have different sensitivity on examined object. Structural attack is similar to visual attack, but it is computer based and it is focused on discovering irregularities in datastructure of cover medium. Every computer data file has its own characteristic structure. Embedding message will leave trace into such structure. The difference between stego file and cover file is given by quality of steganographic tool. Statistical attack has more scientific approach than two mentioned above and it is more

complicated. In general, statistics is used for determinig level of randomness, entropy of the redundant data or color frequencies occurrence in stego files.

Statistical steganalysis has been deeply described by many researchers, e.g. by Niel Provos and Peter Honeyman in [8] or [9]. Andreas Westfeld together with Andreas Pfitzmann introduced their Chi-square statistical attack [10]. Jessica Fridrich and her teams published many research papers on JPEG steganalysis [11], [12], [13] on conventional mathematical – statistical basis. There were more people working on various steganalysis techniques. All above mentioned have been the most dedicated.

However, all techniques described by above papers have been powerfull and functional. They suffer on false positive classification. The reason is simple, the steganography classification is mathematically complicated proces and input steganograms are strongly diversified.

The approach to the steganalysis proposed in this thesis is based on artificial intelligence, mainly artificial neural networks (ANNs). ANNs are known as strong tool for solving difficult classification tasks. ANNs have been successfully implemented in many other projects focused on classification. Artificial intelligence aimed on steganography and steganalysis has been rare [14], [15]. Most of project had involved with Support Vector Machine (SVM), which is a competitive learning method to ANN or other machine learning environment such as Weka [16]. A big challenge for artificial intelligence based on classification was to deal with double compression of JPEG, file which was main source of false positive classification. This thesis and research project is focused on pin-pointing stego images classification by a new sampling methodology and reducing of false positive classification by means of trained ANN classifier on pairs of cover - stego samples.

3 THE OBJECTIVES OF THE DISSERTATION

In my work I would like to continue with research in the field of steganalysis and revealing hidden informations by means of neural networks. All following points should be contained in the thesis.

The steps already done:

- to prove that neural networks are able to do classification of steganograms,
- simulations with different types of stego programmes,
- simulations with different types of neural networks.
- to prepare own decoder of JPEG,
- to test a successful rate of neural network detector against the classic linear classification,
- to try to detect different sizes of hidden message compared to size of cover image
- to try to optimize length of inputs into neural networks and decrease the complexity of neural network.

4 STEGANOGRAPHY IN THEORY

Steganography, as art of hidding information, has been known for over 2500 years. Back then steganography was mainly used for diplomatic, military and a very few people used it for personal purposes along with cryptography. Steganography as well as cryptography have a goal to secure transmited information between the sender and the recipient, but both systems are used in a different way. Cryptography is aimed on transformation of input data into unreadable output. Level of information security depends on the quality of cryptograpic algorithm and correct cipher key selection. Steganography has a different approach, stegomessages also reffered as steganograms are made in such a way that they do not attract attention to themselves. Even transfer remains undetected if steganography is used correctly. No matter how strong cipher can be used, there is always an attempt to wiretape the crypted message and try to break cipher or recover cipher key. However if it is not possible to determine message itself there is nothing to do. The very best solution for securing messages and transport medium is to use cryptography for transforming message into unintelligible gibberish, reffered as ciphertext, and steganography to cover a whole message message along with transport medium.

First documented steganography application was around 440 BC where Demaratus sent a warning about a forthcoming attack to Greece on a wax tablet. The message in that case was written on a wooden backing and then covered by beewax. It appeared as unused. Second one was from that time too. But this time a different transfer medium was chosen. The steganogram was made as a message tattoed on slave's clean shaven head. Then they waited for hair to grow back and then send the slave to deliver the message. The author of the steganogram was called Histiaeus and the purpose was to instigate a revolt against the Persians [18].

Steganography became very popular during Second World War where there was a limited amount of usable communication routes for resistance in Europe that made a perfect envinronment for developing methods of secret communication. Crucial was simplicity information exchange and high level of security. Messages were delivered through radio broadcast coded into birthday wishes, name day wishes or in advertisment

that was the reason why Nazist baned Low frequency radio receivers under the death sentence.

Among other steganography techniques used in Second World War were various kinds of invisible ink or microdots. Microdots are a text or an image substantially reduced in size onto a 1mm disc to prevent detection by unintended recipients. Microdots placed into regular text message would microdot provide a very good transportation medium with perfect protection of transferred secret message. All mentioned methods belong into so called mechanical steganography that can still be used nowadays but as technical development turned 21st century within computer revolution new methods of information transfers have become common for every one. Computer data offers undepletable options for digital steganography.

As mentioned above, steganography has many forms and can be devided into groups by used cover medium and embedding system for secret message.

4.1 Steganography examples divided by cover medium

4.1.1 Physical steganography

4.1.1.1 Hidding one thing inside of another

Basic techniques in physical steganography are e.g. safe place in walking stick, double bottom of carry-on bag or suitcase.

4.1.1.2 Microdots

As mentioned above, microdots are a method used for reducing information into 1mm disk similar to period produced by typewriter.

4.1.1.3 Yellow dots

Yellow dots are produced by color laser printers. Every printed paper is marked by almost invisible code of yellow dots representing printer name, date and time stamp.

4.1.1.4 Code recognition and automatic code extraction from Fujitsu

Barcode is embedded into printed image and remain readable by portable devices [21], steganogram is combination of human understandable information represented by

pictogram and embedded computer readable data (QR code). Information extraction is possible trough cellphone with camera or similar handled device.

4.1.1.5 The letter size, spacing, typeface

The main idea of typography modification is to cover embbeding methodology look like an typing error or unusual document layout. In fact, it is a very good method of message transportation trough public media such as newspapers, magazines, no matter if the text is printed or electronical [22].

4.1.2 Digital steganography

4.1.2.1 Text

Whitespaces – the method is used to conceal messages in ASCII text by appending whitespace to the end of lines. Spaces and tabs are generally not visible in text editors by default. The message is effectively hidden from casual observers.

Text steganography has more techniques of hiding information, for example: Open space methods, Word shifting coding, Line shifting coding, Syntactic methods, Semantic methods, Feature coding [23].

To utilize text steganography has a several reasons. Firstly, a secret message coded into an internet article or email message will not attract any attention when transferred. Another reason may be efford to stylize cover text into specific form eg. SPAM message. Propagation steganography can be used to generate an artificial message with content similar to SPAM message Existence of such message would not attract any attention due to common ocurance. Symantec in their MessageLabs Intelligence: 2010 Annual Security Report [24] announced that more than 89 percent of all email traffic world wide is SPAM. SPAM message can be artificially made by mimic algorithm or mimicry which is an example of propagation steganography and can be use to make artificial texts with look of average internet article or advertisment. Mimic texts are not linguistically correct, but statistically they are good enough to fool spam filters. Mimic is not capable of fooling a human, at least in most cases. If a mimic text is investigated it will strongly support the idea of SPAM message.

Every human language has its own statistical “fingerprint”. Fingerprint is based on frequency analysis of each letter used in a specific language. For English language the letter frequency is shown in Fig. 3. When building a mimic message this knowledge is crucial, The message has to be made in such manner that end product will match this fingerprint. Mimic is useful for sending a big amount of needed data. Using of mimic supports the idea of spamming instead of suspect cover communication transfers.

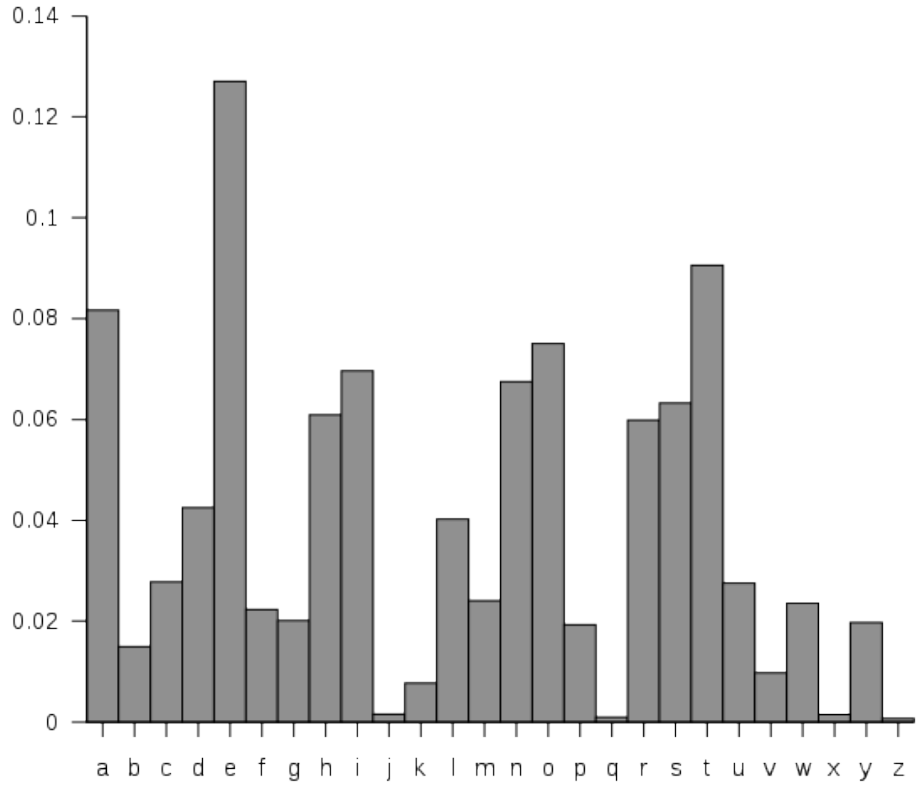


Fig. 3 A typical distribution of letters in English language text [25].

4.1.2.2 Image

Graphic files are the most common data files on the Internet after text information files. Computer graphics is the ideal cover medium for covert communication because of the limitation of the human eye as well as the limited representation of digital technology. There are many different image formats but only a few of them are well discussed in focus to the steganography.

Well suited file for steganography is BMP in its 24-bit color scheme which is possible to hide the content of the whole book without a significant change of the original image information. The most commonly used hiding techniques are: bit insertion, bit deletion, flag bit, threshold bits, direct bit replacement and neighbor parity [26], [27].

Usually the still image represented by a BMP image format has a large size, which is inefficient for transmiting over the Internet. This was the reason for a new type of stego tools for compressed graphic files such as GIF, PNG and JPEG.

JPEG steganography is more complicated then usual still image steganogpraphy because raw image data is not directly accesible as for example on 24 bit RGB BMP model. JPEG steganography is generally based on Least Significant Bit (LSB) applied during the discrete cosine transformation (DCT), information are hidden in the frequency domain. The DCT algorithm is one of the most important components of the JPEG compression.

Further information about JPEG steganography and steganalysis is discussed in this thesis.

4.1.2.3 Audio

Low-bit

Encoding replaces the least significant bit of information in each sampling point with a coded binary string. While this method can be efficiently employed to encode fairly large amounts of hidden data in a given audio signal, it does so at the expense of introducing significant noise at theoretical upper limits [28].

Phase coding

Phase coding works by substituting the phase of an initial audio segment with a reference phase that represents the data. The phase of subsequent segments is adjusted in order to preserve the relative phase between segments [29]. While phase coding is discrete in comparison to low-bit encoding, it is also a more complicated method [28].

Spread spectrum

This method spreads the encoded data across as much of the frequency spectrum as possible, making it difficult for adversaries to find the data, unless they have access to, or can reconstruct the pseudorandom signal used to spread the data across the frequency range [28].

Echo hiding

Echo data hiding embeds data into a host signal by introducing an echo. The data is hidden by varying three parameters of the echo: initial amplitude, decay rate, and offset. As the offset between the original and the echo decreases, the echo blends. In a method similar to vocal overdubbing, where two tracks of vocals are combined to "fatten" up a vocal, this echo blending is perceived as part of the original sound, not an adjunct sound. At some point, the echo and original sound are not perceived as "separate" by the human ear [28].

4.1.2.4 Video

Most of the previously mentioned methods are suitable for various kinds of cover mediums; video files and streams are not exception. Video files are nothing else than large number of images and sound data. Video steganography has an advantage over the static image because of the continuous stream of information and large space for hidden information.

Plane Decomposition Steganography

Plane decomposition steganography uses the bit-plane decomposition. When an image is decomposed into bit-planes, the result is a binary image for each bit-plane. An 8-bit plane (LSB plane) looks as random data but higher planes consist of noise-like regions and informative (not noise-like) regions. It is possible to store hidden information in the noise regions in higher planes. However it will be random-like under the ideal condition as well.

4.1.3 Steganogaphy divided by embedding method

Injection steganography

Injection steganography is an embbeding method used on existing cover media such as text, image, audio file etc. Steganogram is made by inserting a secret message payload into cover media in such manner that it increases its size but does not effect a future presentation of steganogram by original application. Software should not reveal any content devaluation of original data in represented steganogram.

Substitution steganography

Substitution steganography exploits least significant bit to embed information into existing cover media. Embbeding is done trough change of insignificant part of original file for viewer; for example in BMP image file steganography changes information about colour in each pixel. A change of one bit in 24 bit colour depth results in the unrecognizable change of image for human eyes. Substitution steganography is possible to use also on binary files. In this case, information is usualy coded into executable code that is rarely or never used. Steganogram has to survive final processing of original application. It is the same as steganogram is made by means of injection steganography.

Propagation steganography

Propagation steganography is based on generation engine which when fed secret message produce cover medium as know as mimic. Mimic is artifically made "content" and can have many forms. It can be text, image or audio file. Usualy this content has an extraordinary structure without any specific meaning. A good example of this is fractal. Propagation steganography differs from first two noted method. It does not use an external cover (transport) media, only input itneed is secret message optionaly password for encryption. [26].

4.2 Steganography tools used for message embeding

4.2.1 Outguess steganography tool

OutGuess is a universal steganography tool which is able to insert hidden information into redundant bits of input data [32]. Type of input data is not important for

OutGuess at all because the program use specific drivers for specific graphic formats which extract redundant bits and after changes write inside back. The version, which was used for the simulations, is able to work with formats JPEG and PNG. JPEG pictures were used in this work. OutGuess is available under Berkeley Software Distribution (BSD) license. OutGuess is hard to detect by means of statistics calculation based on frequency analasis. Results of statistical analysis are not able to reveal steganography content because OutGuess finds out a maximal length of the message before inserting inside the picture. This causes that result image is not changed from the point of view of frequency analysis as was described in [33].

4.2.2 Steghide steganography tool

Steghide is a steganography program that is able to hide data in various kinds of image- and audio-files. The colour- respectively sample-frequencies are not changed thus making the embedding resistant against first-order statistical tests. Steghide uses a graph theory approach to steganography. The embedding algorithm roughly works as follows: At first, the secret data is compressed and encrypted. Then a sequence of pixel positions in the cover file is created based on a pseudorandom number generator initialized with the passphrase (the secret data will be embedded in the pixels at these positions). These positions that do not need to be changed (because they already contain the correct value by chance) are sorted out. Then a graph-theory matching algorithm finds pairs of positions such that exchanging their values has the effect to embedding of the corresponding part of the secret data. If the algorithm cannot find any more such pairs all exchanges are actually performed. The pixels at the remaining positions (the positions that are not the part of such a pair) are also modified to contain the embedded data (but this is done by overwriting them, not by exchanging them with other pixels). The fact that (most of) the embedding is done by exchanging pixel values implies that the first-order statistics (i.e. how many times a colour occurs in the picture) is not changed. For audio files the algorithm is the same, except that audio samples are used instead of pixels. The default encryption algorithm is Rijndael with a key size of 128 bits (which is AES - the advanced encryption standard) in the cipher block-chaining mode [34].

4.2.3 F5 algorithm - CipherAWT steganography tool

The F5 steganographic algorithm was introduced by German researchers Pfitzmann and Westfeld in 2001 [35]. The goal of their research was to develop concepts and a practical embedding method for JPEG images that would provide high steganographic capacity without sacrificing security. Guided by their χ^2 attack, they challenged the paradigm of replacing bits of information in the cover-image with the secret message while proposing a different paradigm of incrementing image components to embed message bits. Instead of replacing the LSBs of quantized DCT coefficients with the message bits, the absolute value of the coefficient is decreased by one. The authors argue that this type of embedding cannot be detected using their χ^2 statistical attack.

The F5 algorithm embeds message bits into randomly chosen DCT coefficients and employs matrix embedding that minimizes the necessary number of changes to embed a message of certain length. According to the description of the F5 algorithm, version 11, the program accepts five inputs:

- Quality factor of the stego-image Q;
- Input file (TIFF, BMP, JPEG, or GIF);
- Output file name;
- File containing the secret message;
- User password to be used as a seed for PRNG;
- Comment to be inserted in the header. [11]

The F5 algorithm modifies the histogram of DCT coefficients, but some crucial characteristics of the histogram are preserved, such as its monotonicity and monotonicity of increments. The F5 algorithm cannot be detected using the χ^2 attack because the embedding is not based on bit-replacement or exchanging any fixed pairs of values [11].

4.2.4 PQ algorithm / steganography tool

Perturbed quantization (PQ) steganography [5] is a quite successful data hiding approach for which current steganalysis methods fail to work [6]. In other words, PQ does not leave any traces in the form that the current steganalysis methods can catch. However, linear dependency between image rows and/or columns in the spatial domain is affected by PQ embedding due to random modifications on discrete cosine transform (DCT) co-efficients' parities during data hiding.

In PQ steganography, the cover object is applied an information reducing operation that involves quantization such as lossy compression, resizing, or A/D conversion before data embedding. The quantization is perturbed according to a random key for data embedding, therefore called "perturbed quantization." PQ steganography, which uses JPEG compression for information reducing operation, is different from their DCT based counterparts. Since message bits are encoded by changing DCT parities after quantization, the cover image can be thought of just as a recompressed input image. To achieve high embedding rates, recompression is realized by doubling the input quantization table with the assumption that recompression of cover JPEG images does not draw any suspicion because of its wide usage in digital photography [5]. Since the original cover image is recompressed via embedding operation, its compressed version should be considered as "stego" instead of original image.

5 STEGANALYSIS BASICS

Steganalysis is of the same age as steganography itself. During the development it turned into various forms and techniques. This chapter introduces briefly into the art of „hunting ghosts“ or more formely into basic of revealing hidden information.

5.1 Visual attack

Visual steganalysis is one of the most basic forms of steganalysis for digital images. Image is reduced to the single bit plane, most often to the LSB bit plane and then investigated by human eye for any periodic or other type of suspicions patterns in the image. Another approach is to observe colour changes, Some basic algorithms have poor color handling in information embedding process. On other hand, a lot of available steganographic tools provide almost identical output as original file. Although, this is a basic technique for steganalysis. It has to face two major issues of detection. First one is variable sensitivity of human vision among observers, i.e. different condition for classification and secondly this method is not suitable for large volumes of image data.

5.2 Structural attack

Computer information has an internal structure like header, body, pointers etc. It is called file fomat. Every form of document has its own specific format, raw data even a specific structure. The way, how information is made, is crucial to their data structure. It is possible to represent same information with different structure without a change of format. It is similar to musician or singers. There can be two persons singing one song and result is always different. The way of recognition of one singer from another by significant vocal is similar to method how to trace digial data to its source by its structure. Steganographic algorithm usualy leaves behind a characteristic structure of data. Those structures are very commonly used for pattern classification of steganograms.

5.3 Statistical attack

Statistical tests can reveal that an image has been modified by steganography on the basis that an image statistical properties deviate from a norm. Regular statistical tests are independent from the data format and just measure the entropy of the redundant data. It is supposed that images with hidden data have higher entropy than those without.

Westfeld and Pfitzmann observed that embedding encrypted data into a GIF image changes the histogram of its colour frequencies [10]. One property of encrypted data is that the one and the zero bits are equally most likely. When using the least-significant bit method to embed encrypted data into an image that contains colour two more often than colour three, color two is changed more often to colour three than the other way around. As a result, the difference in colour frequency between two and three is reduced by the embedding.

The same is true for JPEG images. Instead of measuring the colour frequencies, it is analyzed the frequency of the discrete cosine transformation (DCT) coefficients. Fig. 4 shows an example where embedding a hidden messages causes noticeable differences to the DCT coefficient histogram.

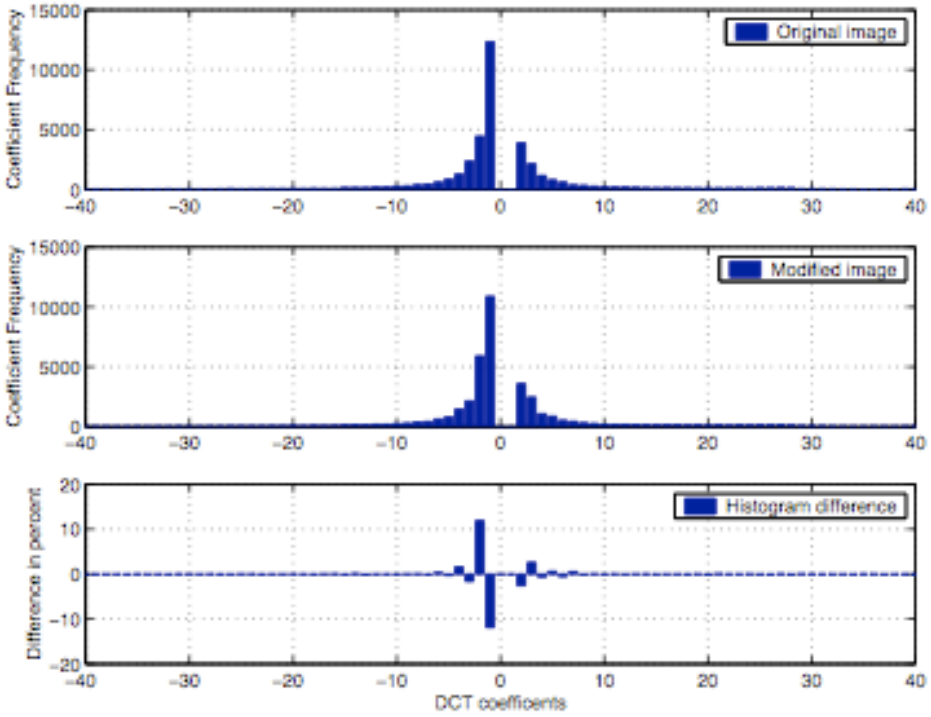


Fig. 4 Example of changes in the histogram of DCT coefficients for original and modified JPEG file, borrowed from[10].

For statistical analysis usualy a χ^2-test is used to determine whether an imageshows distortion from embedding hidden data. Because the test uses only the stego medium, the expected distribution $y_i$for the χ^2-test has to be computed from the image. Let $n_i$be the frequency of DCT coefficients in the image. It is assumed that an image with embedded hidden data has similar frequency for adjacent DCT coefficients. As a result, it can be taken the arithmetic mean (1),

$$y_i^* = \frac{n_{2i} + n_{2i+1}}{2} \tag{1}$$

to determine the expected distribution. The expecteddistribution is compared against the observed distribution (2)

$$y_i = n_{2i} \tag{2}$$

The χ^2 value for the difference between the distributions is given in (3).

,

$$\chi^2 = \sum_{i=1}^{\nu+1} \frac{\left(y_i - y_i^*\right)^2}{y_i^*} \tag{3}$$

where ν are the degrees of freedom, i.e. one less than the number of different categories in the histogram.

The probability p of embedding is then given by the complement of the cumulative distribution function (4).

$$p = 1 - \int_0^{\chi^2} \frac{t^{(\nu-2)/2} e^{-t/2}}{2^{\nu/2}\Gamma(\nu/2)} dt \tag{4}$$

where **Γ** is the Euler Gamma function.

The probability of embedding is computed for different parts of the image. The selection depends onwhat steganographic system is required for. For an image that does not contain any hidden information, the probability of embedding is supposed to be zero everywhere. Fig. 5 and Fig. 6 show the embedding probability for an image without steganographic content and for an image that has hidden content in it.

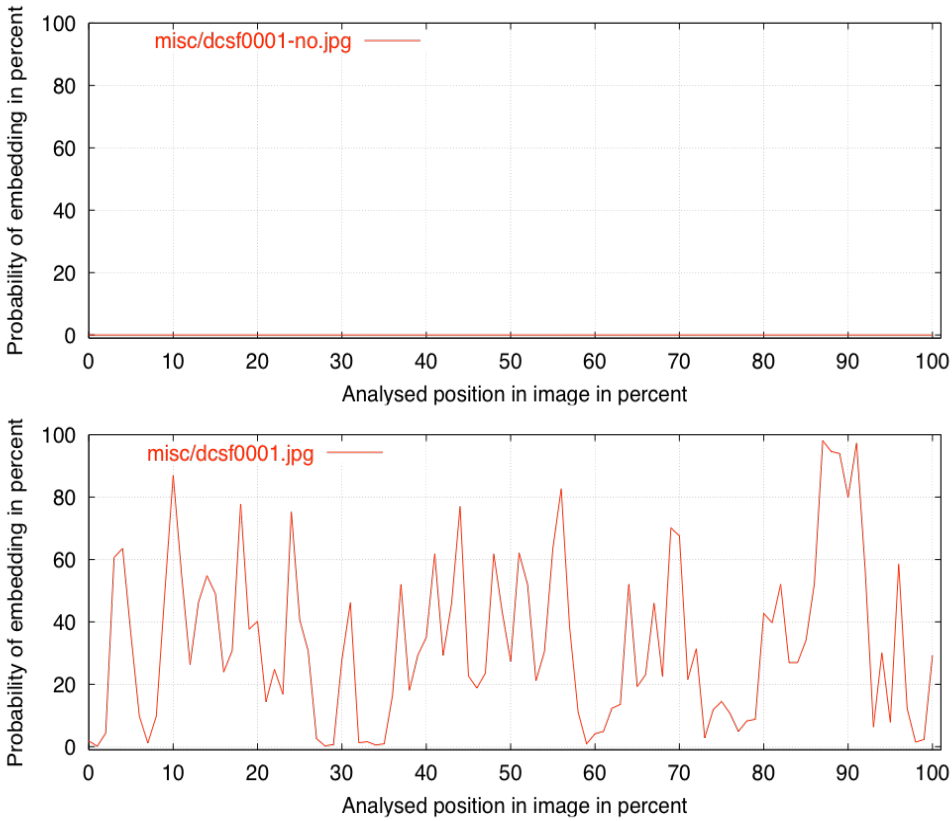


Fig. 5 The probability of embedding calculated for different areas of an image. The upper graph shows the results for an unmodified image, the lower graph shows theresults for an image with steganographic content, borrowed from [10].

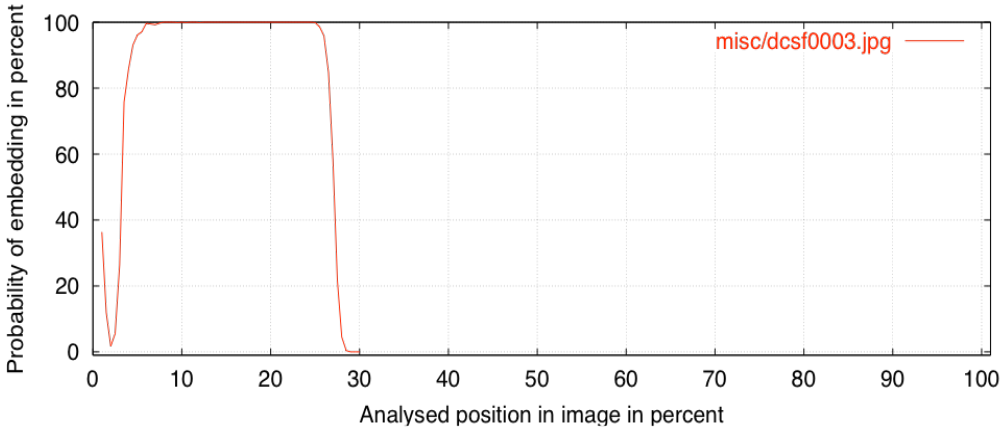


Fig. 6. An image containing a message hidden with JSteg shows a high probability of embedding at the beginning of the image. It flattens to zero, when the test reaches the unmodified part of the DCT coefficients, borrowed from [10].

Several projects aimed on steganalysis have been done over the last 10 years. They were of a different kind to prove that steganograms (e.g. image with a secret message inside) are detectable. Some techniques are described in following chapters.

5.4 Linear discriminant analysis

Probably the best known tool is Stegdetect developed by Niels Provos [38]. On his web site it can be found that Stegdetect is an automated tool for detection of steganographic content in images. It is capable of detecting several different steganographic methods for embedding hidden information in JPEG images. Currently the detectable schemes in Stegdetect v 0.5 are:

- jsteg,
- jphide (unix and windows),
- invisible secrets,
- OutGuess 01.3b,
- F5 (header analysis),

- appendX and camouflage.

The whole detection mechanism (Fig. 7) is based on a linear discriminant analysis. Stegdetect can automatically determine a linear detection function on the basis of given a set of normal images and a set of images that contain a hidden content. Linear discriminant analysis computes a dividing hyperplane that separates the non stego images from the stego images. The hyperplane is characterized as a linear function. The learned linear detection function can be applied later to yet unclassified images.

Stegdetect supports several different feature vectors and automatically computes receiver operating characteristic which can be used to evaluate the quality of the automatically learned detection function [38].

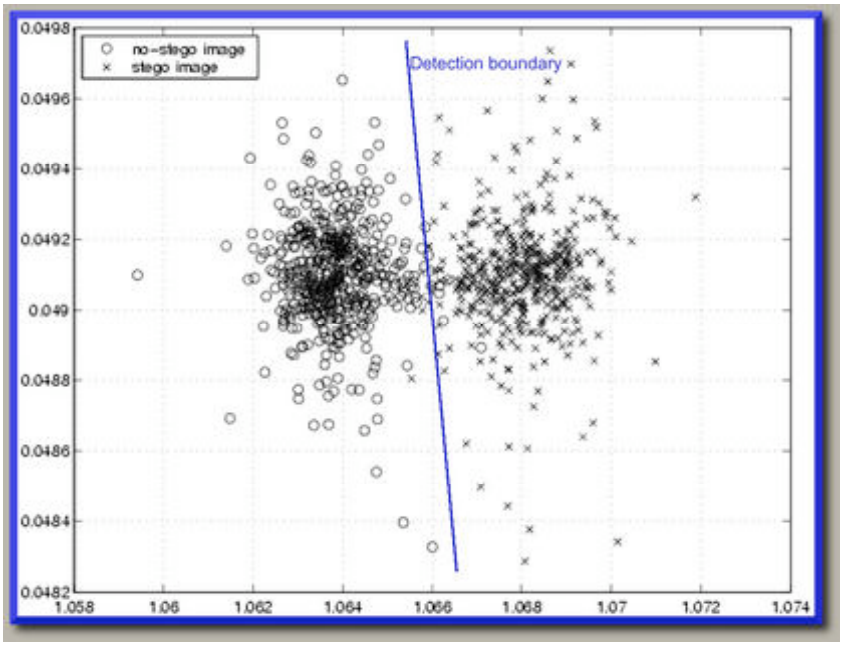


Fig. 7. Linear discriminant analysis, borrowed from[38]

6 IMAGE PREPROCESSING

In order to make accurate steganalysis tool, it is necesary to discover changes made by steganographic tools to the JPEG images. This chapter is dedicated to illustration of image changes caused by embedding information in to cover images with steganography tools: Outguess, Steghide, F5 algorithm in CipherAWT, PQ and method of data sampling for future analysis by neural networks.

6.1 Illustration how steganography effects JPEG file

Illustration of changes made by steganography is not easy task at all. If it would be there is no interest in steganalysis. In next paragraphs, set of images are depicted for each steganographic tool or algorithm divided in matrix 2x2 as shown in Table 1.

Table 1 - Position of images in matrix

A	B
C	D

where image position A is a cover file; B is a stego image with 600 byte encoded message. Pixel to pixel analysis was made on images C and D. Such an analysis works so that a cover image is taken and analyzed pixel by pixel with matching pixel in stego image. If any change was found, this pixel was set to black colour in output image, otherwise the pixel was set to white. By this output, image represents unchanged pixels by white colour and changed by black colour. Two different message lenghts were used. Picture C carries 30 bytes long message and D image carries 600 bytes long message. This does not mean that black marked pixels carry a hidden message it only represents manipulation in this particular pixel which could be done because of image optimalization for size of secret message to fit the cover image.

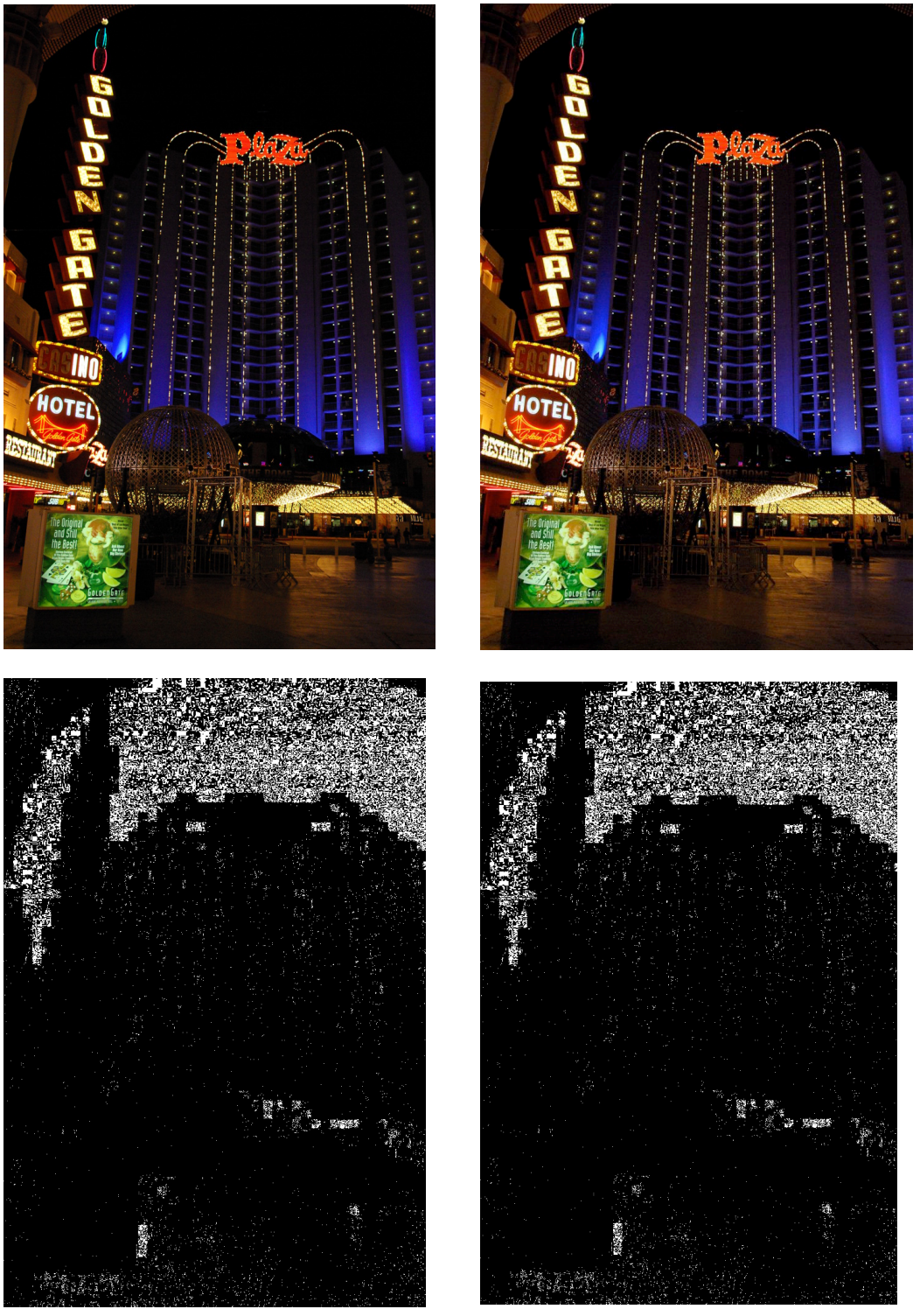


Fig. 8 Example of F5: A - cover, B - stego , C - diff 30 B msg. , D - diff 600 B msg.

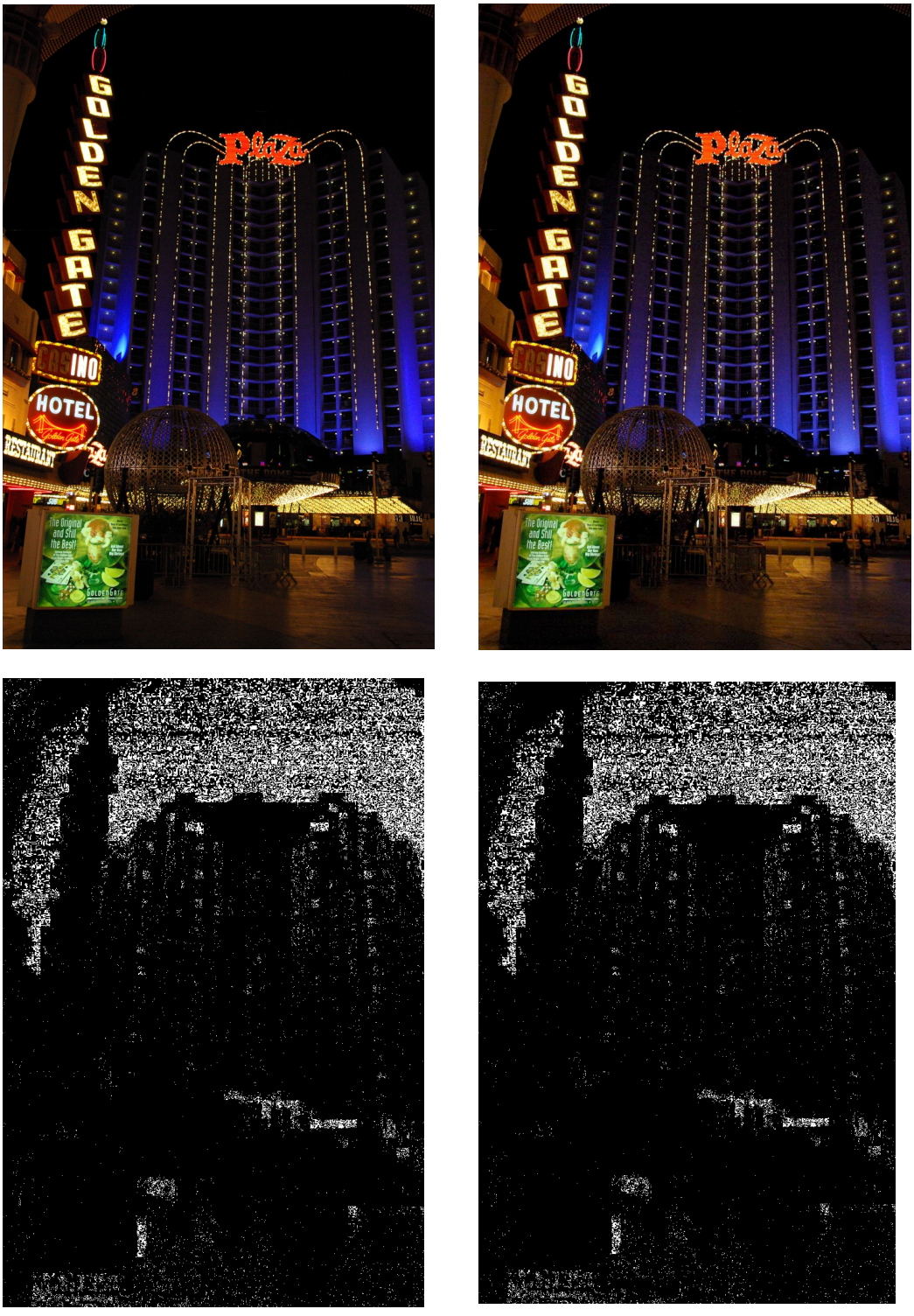


Fig. 9 Example of Outguess: A - cover, B - stego , C - diff 30B msg. , D - diff 600 B msg.

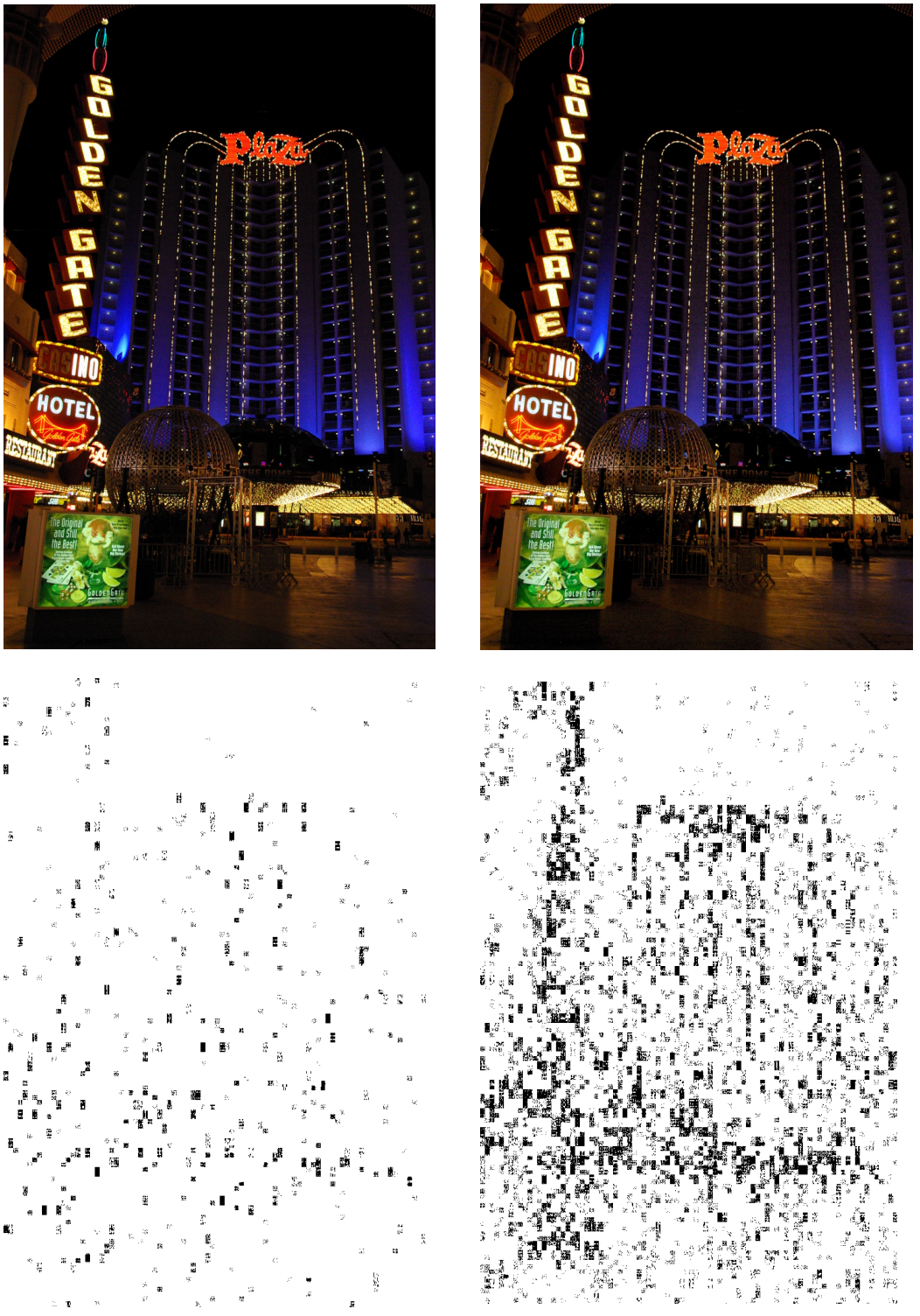


Fig. 10 Example of Steghide: A - cover, B - stego , C - diff 30 B msg. , D - diff 600 B msg.

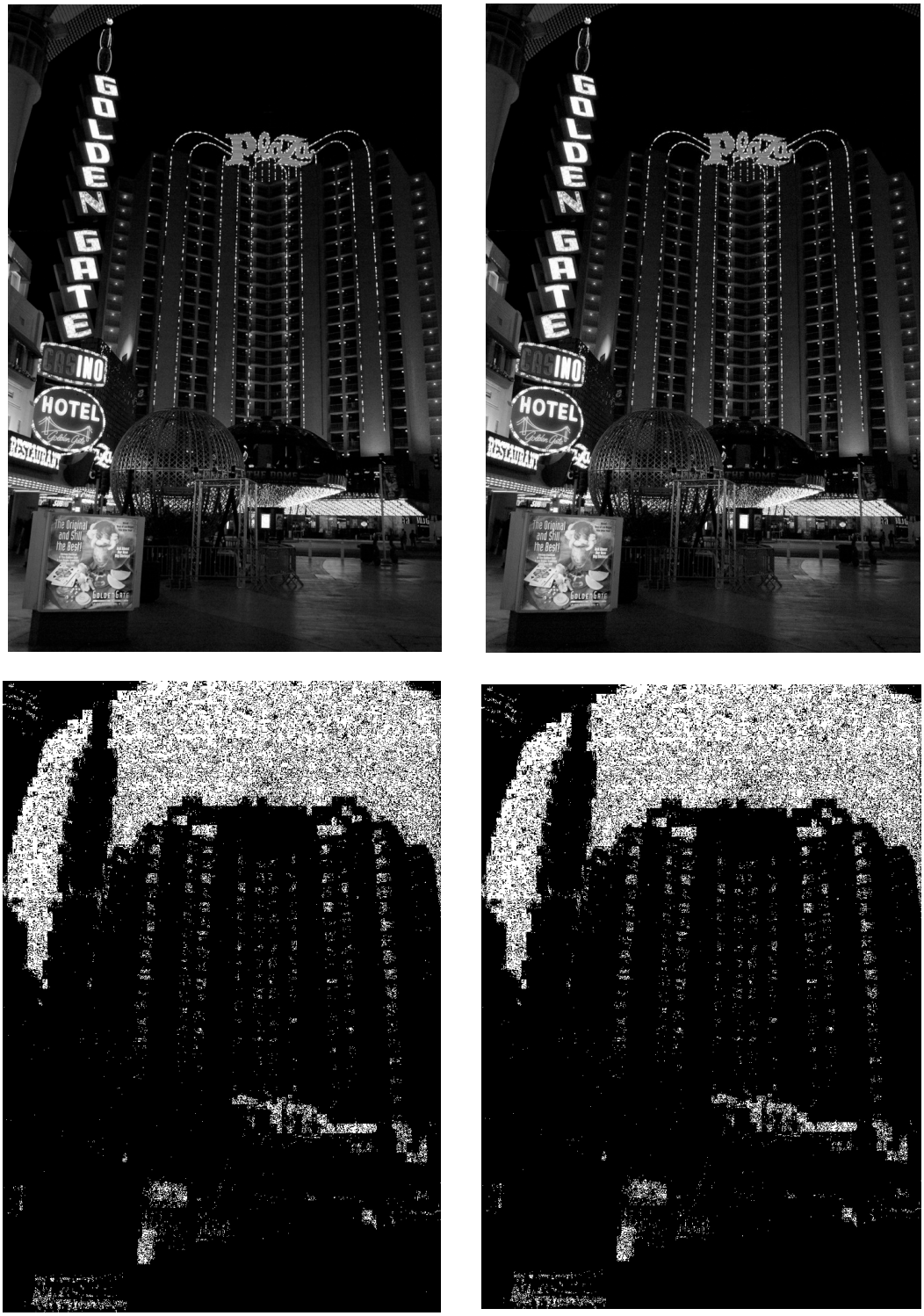


Fig. 11 Example of PQ: A - cover B - stego , C - diff 30 B msg. , D - diff 600 B msg.

6.2 Quantization tables

First significant changes discovered during the analysis of JPEG steganograms have been found in quantization tables. Values of quantization tables are used for dividing values of DCT coefficients which is done to reduce amount of information in high frequency components. Analysis of values of quantization tables showed significant differences between stego and cover images. However, the statistical analysis of the image samples had shown an insufficient amount of divergency among samples of the same class. Thus this sampling method could not be use for a training set for neural networks.

6.3 Huffman code

Quantization tables were dead end, for steganalysis by means of artificial neural networks because the changes were too small amongst the cover and stego images. Therefore a need for some other tool which gives an information about the inner structure arises. Choice was the Huffman coding. Huffman coding in JPEG works as a background for lossless compression.

Huffman coding was designed by David Huffman in 1952. This method takes symbols represented e.g. by values of discrete cosine transformation (which is one of methods how to present information in pictures like colour, brightness etc.) and coded it into changeable length code so that according statistics the shortest bit representation to symbols with the most often appearance. It has two very important properties – it is a code with minimal length and prefix code that means that it can be decode uniquely. On the other hand, the disadvantage is that we must know appearance of each symbol a priori. But in the case of pictures we can work with estimation, which will be edited during the compression [39] . Fig. 12, Fig. 13 and Table 2 show the differences between cover and stego images in DC or AC (direct or alternating part) class. The pictures show number of each bit word in the image.

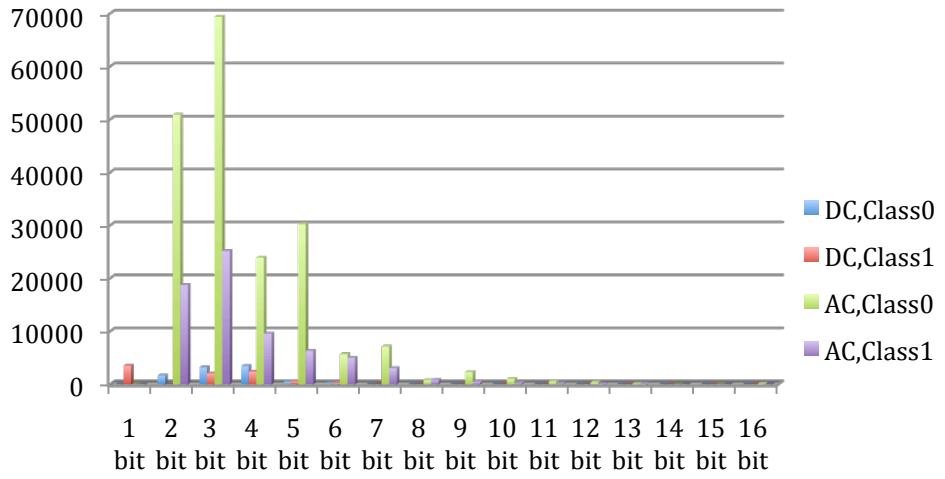


Fig. 12. Graph of Huffman coding histogram – cover image (clear picture)

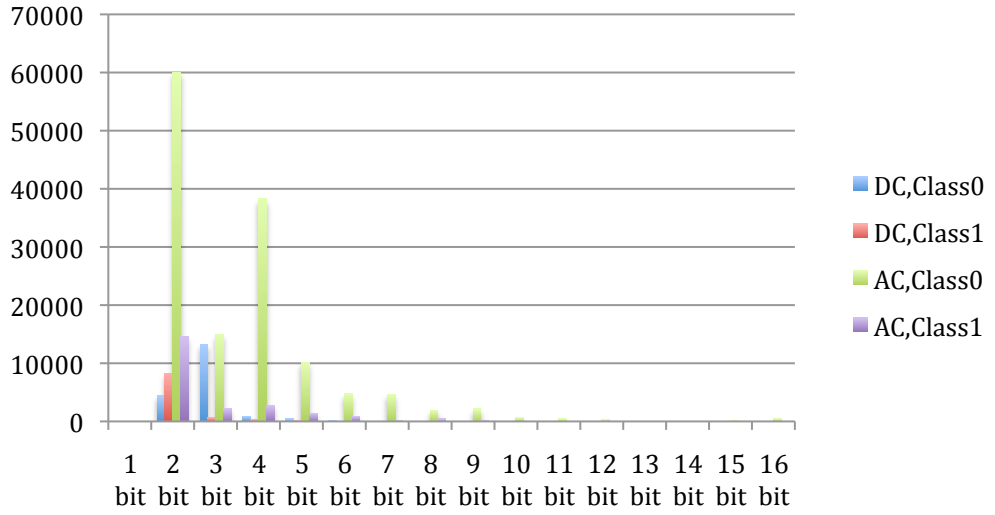


Fig. 13. Graph of Huffman coding histogram – stego-image (coded picture)

Table 2. Graph of Huffman coding histogram – a) cover image, b) stego image

a)

	DC, Class0	DC, Class1	AC, Class0	AC, Class1
1 bit	0	3504	0	0
2 bit	1623	0	50871	18704
3 bit	3178	2060	69370	25155
4 bit	3435	2371	23902	9522
5 bit	342	527	30216	6311
6 bit	86	170	5642	4968
7 bit	0	31	7102	3032
8 bit	0	1	771	805
9 bit	0	0	2285	425
10 bit	0	0	1022	204
11 bit	0	0	522	115
12 bit	0	0	345	40
13 bit	0	0	74	49
14 bit	0	0	20	8
15 bit	0	0	0	6
16 bit	0	0	50	13

b)

	DC, Class0	DC, Class1	AC, Class0	AC, Class1
1 bit	0	0	0	0
2 bit	4433	8312	59998	14595
3 bit	13283	730	14904	2224
4 bit	906	343	38276	2755
5 bit	444	181	10142	1444
6 bit	86	10	4742	925
7 bit	0	0	4680	89
8 bit	0	0	1943	428
9 bit	0	0	2149	77
10 bit	0	0	667	42
11 bit	0	0	444	12
12 bit	0	0	316	0
13 bit	0	0	0	0
14 bit	0	0	0	1
15 bit	0	0	73	0
16 bit	0	0	477	0

To imagine more how it works two foollowing pictures (Fig. 14 a) and b)) can be used. Each bit word can stand as a brick in the wall. It is possible to get two same big walls but each one will be assembled from different bricks and brick sizes. These two walls are of the same size but of different structure (different set of bricks, some bricks appear more often then others). By same analogy, differences in cover and stego files can be viewed. The aim is to compare the different bit word length and different sizes of bricks in the walls for cover and images affected by steganography.

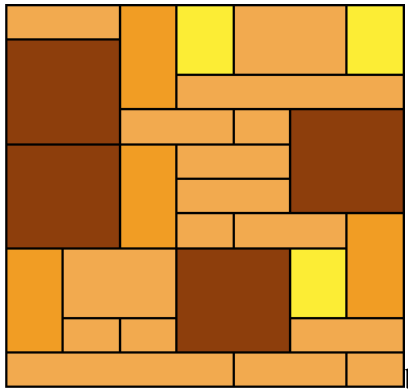
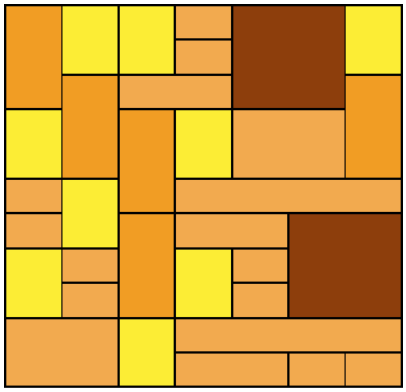
a)  b) 

Fig. 14. Illustration of Huffman coding histogram – a) cover image, b) stego image

The main goal of steganography is not to attract attention stego images appear as usual pictures taken by digital camera. But there are significant changes in the structure of stego images. These changes in JPEG structure are relevant and used in this case for correct training of artificial neural network.

7 NEURAL NETWORKS

Artificial neural networks (ANN) are tools of artificial intelligence which were developed in first half of 1940s. After Pitts – McCulloch model [40] of neuron (Fig. 15) and Rosenblatt's first neural net Perceptron with learning algorithm were published, Minsky and Papert caused the leaving of ANN for some time because the Perceptron was not able to solve nonlinear separable problems. Fortunately, in 1980s researches came back and a boom has started [3].

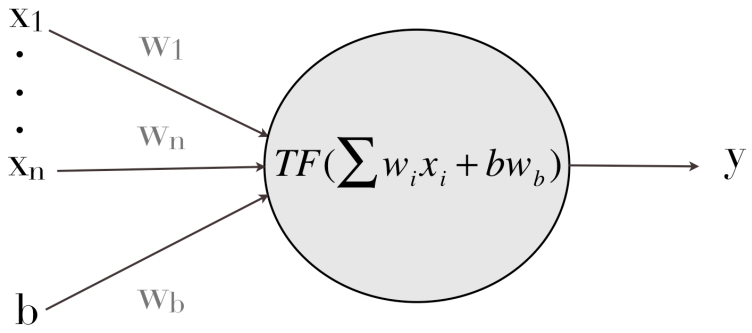


Fig. 15. Model of neuron – TF (transfer function), x_1 - x_n (inputs to neural network), b – bias, $w_1 – w_n$, w_b – weights, y – output

Artificial neural networks are inspired in the biological neural nets and are used for complex and difficult tasks. The most often usage is classification of objects because ANN are capable of generalization and hence the classification is natural for them. Some other possibilities are in pattern recognition, control, filtering of signals and also data approximation. Classification is the property used here.

There are several types of artificial neural networks. Mainly, they are divided on supervised and unsupervised neural networks. Supervised neural nets needs a training set with inputs and required outputs which help to train the neural network. Unsupervised neural networks work on a different basis. They try to group items in training set according similar properties. The other difference is in settings of layers, neurons in layers, types of transfer functions etc. In the case of this thesis, supervised artificial neural nets were used.

Simulations were performed with feedforward net with supervision. ANN needs a training set of known solutions to be trained on them. The neural network works so that suitable inputs in numbers have to be given on the input vector. These inputs are multiplied by weights which are adjusted during the training. In the neuron the sum of inputs multiplied by weights are transferred through mathematical function like sigmoid, saturated linear (Fig. 16), hyperbolic tangent, radial basis functions etc. Therefore ANN can be used also for data approximation.

Feedforward nets have different training algorithms the most known Backpropagation, Pruning algorithm, gradient methods, Levenberg-Marquardt [41] and others. In the performed simulations Levenberg-Marquardt algorithm was used. In future we suppose to use also evolutionary optimization algorithms like Self Organizing Migrating Algorithm (SOMA) or Differential Evolution (DE) [43], [44] because it is supposed that classification of each stego programme could be difficult optimization problem to find suitable weights in neural networks.

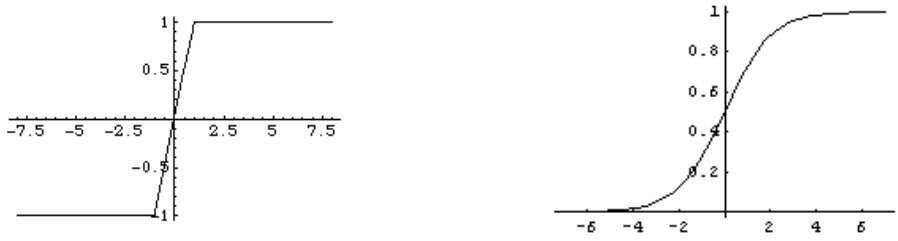


Fig. 16. Linear saturated function (left), Sigmoid function (right)

7.1 Neural networks topologies used in performed simulations

The single neuron units (Fig. 15) are connected to different structures to obtain ANN (e.g. Fig. 17). These networks were design for different tasks. Fig. 18 shows two hidden layer net where the last neuron in the left input layer is bias equal to one.

7.2 FeedForward network with one hidden layer

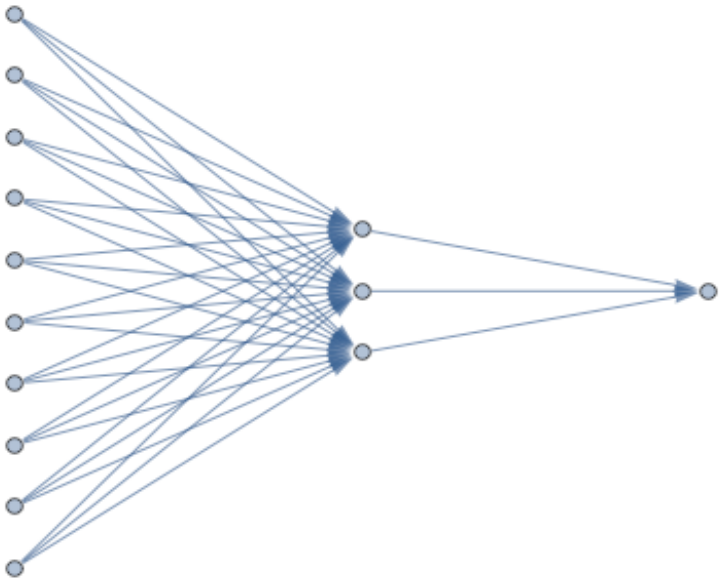


Fig. 17. One hidden layer neural net and one output

7.2.1 FeedForward network with two hidden layers

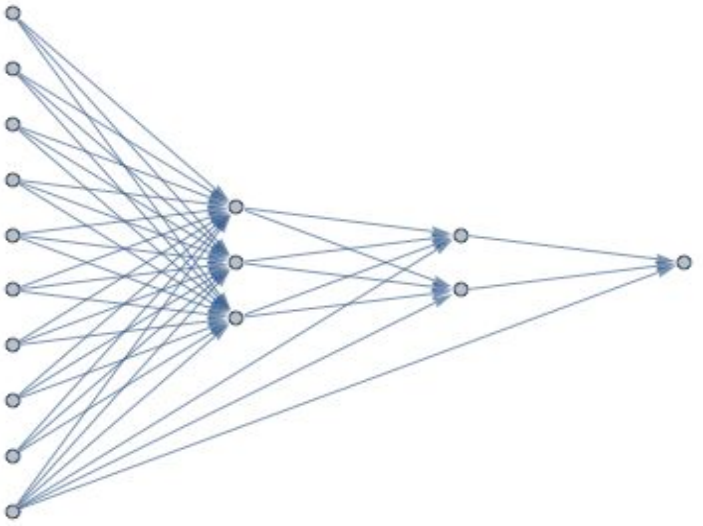


Fig. 18. Two hidden layer neural net

8 EXPERIMENTAL PART

8.1 Preparation of training sets

Firstly, it was necessary to prepare a suitable training set for artificial neural networks. This training set consists of numbers obtained from Huffman coding. Huffman coding was applied on adjustments and modifications of basic 2183 images which have been acqired from three digital cameras (Sony DSC-P93, Olympus SP550UZ, Pentax K10D) in fine or superfine quality for testing purposes. The lowest image resolution for this test group was more then 2560x1600, the average picture resolution is 3529x2458 pixels and maximum picture resolution is 3872x2592 pixels with average file size 2616.6 kB and maximum file size 4403.2 kB.

8.2 Cover samples

Cover samples, i.e. images without any hidden information have been created by resizing original digital images by linux tool ImageMagick [46] into different file resolution estimated by common appearance on internet. Full image pool has almost 22 000 images.

List of all picture resolution used for test group :

800x600, 1024x768, 1280x1024, 1440x900, 1680x1050, 1920x1440, 2560x1600 and one special group containing original files with resolution higher then 2560x1600 pixels..

8.3 Stego samples

All samples from cover image pool were used for Outguess, Steghide and PQ algorithm. Due to problems with F5 java implementation, input cover file pool was reduced only to images up to maximum resolution of 1680 x 1050 pixels in this case.

Secret message and encryption password was generated by linux random number generator that colect environmental noise from device drivers and other sources into entropy pool. The amount of hidden information was set up by measurement of common length of short messages.

List of all message lenghts used for stego test samples:

5, 10, 15, 30, 75, 150, 300 and 600 Bytes.

Training set has been assambled trough a program coded as part of Jiri Sedlak's diploma thesis Processing digital image information for the steganalysis using neural networks [47]. Program has been designed during process of optimization and acceleration of image analysis for purposes of this dissertation. Program is written in Python language. It is optimalized for speed, image handling has been coded from a scratch and it does not require any external python libraries.

Values obtained from Huffman coding were transfered into training set, i.e. all four columns (Table 2) were given column by column to create a training vector. As first simulations discovered that proportional numbers are not suitable further tests used real numbers. The proportional number changes between cover and stego were too small for discovering by means of artificial neural networks. Examples of cover and stego inputs in a training set are depicted in Fig. 19 and Fig. 20. It is a matrix of 2 individual inputs of length 64. These inputs are then given on the input layer of neural network. Output was 0 in the case of cover image and 1 in the case of stego image in a training phase. The attempt to create a universal detector "at once" was not successful. The training error was too high. Therefore the detector checks all stego programs (algorithms) one by one. In such a case just one output neuron is enough to decide if the image is stego or cover.

{{0,2181,49638,11923,7754,3614,2113,1328,181,0,0,0,0,0,0,0,0,37838,17016,9603,6323,4489,2770,691,2,0,0,0,0,0,0,0,0,1184544,266832,407253,225804,86439,84594,39260,27220,14603,6757,1326,0,0,57,5077,0,294505,156163,135414,76507,49051,12597,16388,9120,3521,1513,93,0,881,629,86},

{0,3508,60281,8977,3793,1551,523,99,0,0,0,0,0,0,0,0,37958,19066,13402,5052,1900,1048,252,54,0,0,0,0,0,0,0,0,1289147,134604,445306,266998,147454,78470,41055,33866,12949,7305,1311,0,0,95,3138,0,363928,193743,169972,74293,52812,6726,20723,4739,2088,500,59,0,203,147,82}}

Fig. 19. Example of cover image inputs in a training set – real number case

{{0,2178,49642,11918,7758,3614,2113,1328,181,0,0,0,0,0,0,0,0,37824,17026,9608,6323,4486,2771,692,2,0,0,0,0,0,0,0,0,1184565,266816,406818,225770,85887,84320,39638,27400,14811,6889,1516,0,0,105,5231,0,295156,155514,135282,76214,48989,12495,16659,9154,3609,1601,94,0,868,625,208},

{0,3509,60280,8977,3794,1552,521,99,0,0,0,0,0,0,0,0,37930,19084,13405,5058,1900,1049,252

,54,0,0,0,0,0,0,0,0,1288890,134861,444337,267085,146488,78073,41737,34275,13252,7546,1626,0
,0,180,3348,0,365181,192363,170021,74091,52756,6693,20933,4760,2138,530,61,0,200,147,141}}

Fig. 20. Example of stego image inputs in a training set – real number case

9 RESULTS

Several experiments were carried out during all testing. Firstly, a steganography tool OutGuess for coding a message in and training sets with proportional values of Huffman coding were used in neural networks with one hidden layer net, two hidden layer net and RBF (radial basis function in neurons) net. The results were almost 100% successful as shown in [48]. However Steghide was tried with similar conditions it pointed that this solution is not suitable for usage of proportional values. The small differences in Huffman coding values were not visible for ANN in this case. Therefore for futher simulations training sets with real numbers of Huffman coding as in (Fig. 19) and (Fig. 20) were performed.

The four algorithms were used for classification alone and some also together with different settings of artificial neural networks. The final test was carried out for each stego algorithm alone with 9 combinations of transfer functions in hidden layer and output neuron to find out which possibility is the best one. The following chapters show the preliminary tests with some results. The final test is in a separate chapter.

9.1 OutGuess

6000 saples were used for training and for testing 2196 samples for testing. Results show that used neural networks have almost 100% success. The better was the net with two hidden layers which could be caused by higher number of weights.

The following picture shows an example of training error – root mean square error (RMSE) for two hidden layer net.

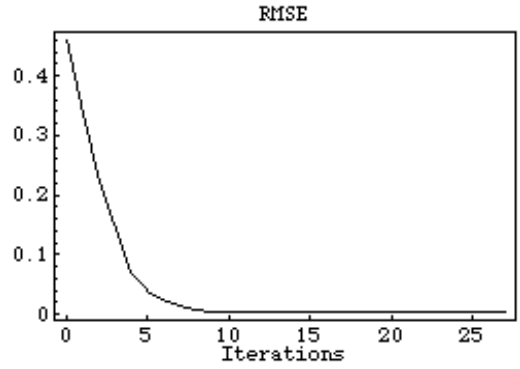


Fig. 1. Root Mean Square Error – RMSE for two hidden layer net

Following tables Table 3 and Table 4) show results for one and two hidden layers and stegoimages coded by OutGuess. As can be seen almost 100 % succes has been reached.

Table 3: Results of detection by one hidden layer neural net – A – Nr. of bad classified cases (mistakes) out of x, B – proportional mistake, C – proportional success (100 – proportional mistake)

	A	*B*	*C*
Cover (clear) *x = 2 196*	*8*	*0.728597*	*99.2714*
Stego (coded) *x = 2 196*	*0*	*0*	*100*

Table 4: Results of detection by two hidden layer neural net – A – Nr. of bad classified cases (mistakes) out of x, B – percentual mistake, C – percentual success (100 – percentual mistake)

	A	*B*	*C*
Cover (clear) *x = 2 196*	*1*	*0.0910747*	*99.9089*
Stego (coded) *x = 2 196*	*0*	*0*	*100*

9.2 Steghide

Firstly, the size of the message was 128 bytes as in the case of OutGuess. But results were not satisfactory. The training error (global root mean square error (global RMSE)) was around 0.3. This value was quite high because the error was around 25 % of incorrect classified data, as it can be seen from following (Table 5 and Table 6).

Table 5:Results of detection by one hidden layer neural net – A – Nr. of bad classified cases (mistakes) out of x, B – percentual mistake, C – proportional success (100 – proportional mistake)

	A	B	C
Cover (clear) x = 2 196	83	7.5592	92.4408
Stego (coded) x = 2 196	284	25.8652	74.1348

Table 6: Results of detection by two hidden layer neural net – A – Nr. of bad classified cases (mistakes) out of x, B – proportional mistake, C – proportional success (100 – proportional mistake)

	A	B	C
Cover (clear) x = 2 196	130	11.8397	88.1603
Stego (coded) x = 2 196	244	22.2222	77.7778

The coded message was in this case 128 bytes, which is between 0.000625 % and 0.0104 % of the image size. Therefore the length of a secret message was increased to interval between 0.4 % and 6 % of the image size dependent on the resolution and the size of the image. Normally, the secure threshold can be around 20 % of the image size. Hence, the length was increased significantly, but still it is very low under the threshold.

The same set of testing images was used as before. And 100 % success was obtained as following tables show.

Table 7: Results of detection by one hidden layer neural net – A – Nr. of bad classified cases (mistakes) out of x, B – proportional mistake, C – proportional success (100 – proportional mistake)

	A	B	C
Cover (clear) x = 2 196	0	0	100
Stego (coded) x = 2 196	0	0	100

Table 8: Results of detection by two hidden layer neural net – A – Nr. of bad classified cases (mistakes) out of x, B – proportional mistake, C – proportional success (100 – proportional mistake)

	A	B	C
Cover (clear) x = 2 196	0	0	100
Stego (coded) x = 2 196	0	0	100

The results show that if a small length of the message is used, neural network has problems to detect it. But usually in real world the message does not contain only one word and if the message is longer but still very low under the secure threshold, neural networks classify the coded and clear messages with a 100% success. In the case of Steghide used messages have less then 20% length of cover images capacity that a secure threshold is. The values were between 0.4 and 6%.

9.3 OutGuess and Steghide

This chapter is concerning both stego programmes – OutGuess and Steghide. Both were used in training set as a stego (coded) picture. In this case the exact name of the programme was not detected, only if a picture had a hidden content or not. Following tables (Table 9 and

Table 10) show results for OutGuess and Steghide together.

Table 9: Results of detection by one hidden layer neural net – A – Nr. of bad classified cases (mistakes) out of x, B – proportional mistake, C – proportional success (100 – proportional mistake)

	A	B	C
Cover - clear (x = 4 068)	1	0.02	99.98
Stego - coded by OutGuess (x = 3 644)	0	0	100
Stego - coded by Steghide (x = 9 933)	0	0	100

Table 10: Results of detection by two hidden layer neural net – A – Nr. of bad classified cases (mistakes) out of x, B – proportional mistake, C – proportional success (100 – proportional mistake)

	A	B	C
Cover - clear (x = 4 068)	809	19.88	88.12
Stego - coded by OutGuess (x = 3 644)	0	0	100
Stego - coded by Steghide (x = 9 933)	0	0	100

These results show that the task starts to be more complex and that simpler neural net with one hidden layer was better, almost 100% successful, than with two hidden layers. Therefore, for futher simulations only one hidden layer neural net was used.

9.4 CipherAWT (F5 algorithm)

CipherAWT was carried out with a total of 18000 samples (10 800 cover-images, 7200 F5 stego-images) for training. For testing total of 6777 (4068 cover-images, 2709 F5 stego-images) were performed. Results showed almost 100% success as in above simulations (Table 11).

Table 11: Results of detection by one hidden layer neural net – A – Nr. of bad classified cases (mistakes) out of x, B – proportional mistake, C – proportional success (100 – proportional mistake)

	A	B	C
Cover - clear (x = 4 068)	22	0.54	99.46
Stego - coded by CipherAWT (x = 2 709)	0	0	100

9.5 OutGuess, Steghide and CipherAWT (F5 algorithm)

Also all three algorithms were tested together with following results (Table 12).

Table 12: Results of detection by one hidden layer neural net – A – Nr. of bad classified cases (mistakes) out of x, B – proportional mistake, C – proportional success (100 – proportional mistake)

	A	B	C
Cover - clear (x = 4 068)	0	0	100
Stego - coded by OutGuess (x = 3 644)	0	0	100
Stego - coded by Steghide (x = 9 933)	0	0	100
Stego - coded by CipherAWT (x = 2 709)	0	0	100

9.6 Benchmark test – Stegdetect versus ANN in F5 algorithm

This chapter is focused on comparison of detection between neural networks and Stegdetect on the same set of samples. Neural network was the same as in [49] based on classification of Huffman coding [39] of images. Image preprocessing (to obtain data suitable for neural network) is done by means of a python implementation of JPEG decoder written from scratch.[47] In this case neural network was trained only on the algorithm F5. Two following subsection contain tables with results. Firstly, section 9.6.1 (Table 13 - Table 21) is concerned to Stegdetect tool and section 9.6.2 (Table 22 - Table 30) shows tables for neural networks. Each table figures results for different image resolution and different message payload. At the end of sections, there are tables also for cover images which give results for more resolutions. As described above, Java

implementation of F5 algorithm works only at resolution 1680x1050 pixels but cover images have no limitation. The gray rows show total success and error rate of stego and cover images in the classification (nr. of missclassified samples in stego plus nr. of missclassified samples in cover divided by the total nr. of samples for the resolution).

9.6.1 Stegdetect results

Table 13 – Stegdetect, detecion of 5 byte message

Message 5 bytes	800 x 600	1024 x 768	1280 x 1024	1440 x 900	1680 x 1050
Samples total	2183	2182	2183	2183	2183
Correct classificcaiton	2060	2081	2087	2098	2111
Missclassification	123	102	96	85	72
Success rate in %	94.4	95.3	95.6	96.1	96.7
Error rate in %	5.6	4.7	4.4	3.9	3.3
Total success rate in %	92.9	93.3	93.5	93.7	94.0
Total error rate in %	7.1	6.7	6.5	6.3	6.0

Table 14 - Stegdetect, detecion of 10 byte message

Message 10 bytes	800 x 600	1024 x 768	1280 x 1024	1440 x 900	1680 x 1050
Samples total	2182	2183	2183	2183	2183
Correct classificcaiton	2056	2083	2085	2095	2106
Missclassification	126	100	98	88	77
Success rate in %	94.2	95.4	95.5	96.0	96.5
Error rate in %	5.8	4.6	4.5	4.0	3.5
Total success rate in %	92.8	94.5	96.3	96.8	97.7
Total error rate in %	7.2	5.5	3.7	3.2	2.3

Table 15 - Stegdetect, detecion of 15 byte message

Message 15 bytes	800 x 600	1024 x 768	1280 x 1024	1440 x 900	1680 x 1050
Samples total	2183	2182	2183	2182	2180
Correct classificcaiton	2062	2085	2089	2098	2104
Missclassification	121	97	94	84	76
Success rate in %	94.5	95.6	95.7	96.2	96.5
Error rate in %	5.5	4.4	4.3	3.8	3.5
Total success rate in %	92.9	94.6	96.4	96.9	97.7
Total error rate in %	7.1	5.4	3.6	3.1	2.3

Table 16 - Stegdetect, detecion of 30 byte message

Message 30 bytes	800 x 600	1024 x 768	1280 x 1024	1440 x 900	1680 x 1050
Samples total	2183	2182	2183	2183	2181
Correct classificcaiton	2053	2075	2088	2100	2106
Missclassification	130	107	95	83	75
Success rate in %	94.0	95.1	95.6	96.2	96.6
Error rate in %	6.0	4.9	4.4	3.8	3.4
Total success rate in %	92.9	94.3	96.4	96.9	97.8
Total error rate in %	7.1	5.7	3.6	3.1	2.2

Table 17 - Stegdetect, detecion of 75 byte message

Message 75 bytes	800 x 600	1024 x 768	1280 x 1024	1440 x 900	1680 x 1050
Samples total	2182	2183	2183	2183	2183
Correct classificcaiton	2056	2080	2085	2096	2107
Missclassification	126	103	98	87	76
Success rate in %	94.2	95.3	95.5	96.0	96.5
Error rate in %	5.8	4.7	4.5	4.0	3.5
Total success rate in %	92.8	94.4	96.3	96.8	97.7
Total error rate in %	7.2	5.6	3.7	3.2	2.3

Table 18 - Stegdetect, detecion of 5 byte message

Message 150 bytes	800 x 600	1024 x 768	1280 x 1024	1440 x 900	1680 x 1050
Samples total	2183	2183	2183	2183	2182
Correct classificcaiton	2045	2088	2084	2098	2101
Missclassification	138	95	99	85	81
Success rate in %	93.7	95.6	95.5	96.1	96.3
Error rate in %	6.3	4.4	4.5	3.9	3.7
Total success rate in %	92.5	94.6	96.3	96.9	97.6
Total error rate in %	7.5	5.4	3.7	3.1	2.4

Table 19 - Stegdetect, detecion of 300 byte message

Message 300 bytes	800 x 600	1024 x 768	1280 x 1024	1440 x 900	1680 x 1050
Samples total	2183	2183	2183	2183	2183
Correct classificcaiton	2051	2077	2078	2094	2102
Missclassification	132	106	105	89	81
Success rate in %	94.0	95.1	95.2	95.9	96.3
Error rate in %	6.0	4.9	4.8	4.1	3.7
Total success rate in %	92.6	94.4	96.2	96.8	97.6
Total error rate in %	7.4	5.6	3.8	3.2	2.4

Table 20 - Stegdetect, detecion of 600 byte message

Message 600 bytes	800 x 600	1024 x 768	1280 x 1024	1440 x 900	1680 x 1050
Samples total	2183	2183	2183	2183	2183
Correct classificcaiton	2049	2078	2082	2092	2107
Missclassification	134	105	101	91	76
Success rate in %	93.9	95.2	95.4	95.8	96.5
Error rate in %	6.1	4.8	4.6	4.2	3.5
Total success rate in %	92.6	94.4	96.2	96.7	97.7
Total error rate in %	7.4	5.6	3.8	3.3	2.3

Table 21 - Stegdetect, classification of cover images

COVER images	800 x 600	1024 x 768	1280 x 1024	1440 x 900	1680 x 1050	1920 x 1200	1920 x 1440	2560 x 1600
Samples total	2182	2180	2182	2179	2182	2182	2182	2154
Correct classificcaiton	1993	2040	2119	2027	2159	2170	2181	2153
Missclassification	189	140	63	52	23	12	1	1
Success rate in %	91.3	93.6	97.1	97.6	98.9	99.5	99.95	99.95
Error rate in %	8.7	6.4	2.9	2.4	1.1	0.5	0.05	0.05

9.6.2 Artificial neural network results

Table 22 – ANN, detecion of 5 byte message

Message 5 bytes	800 x 600	1024 x 768	1280 x 1024	1440 x 900	1680 x 1050
Samples total	2183	2182	2183	2183	2183
Correct classificcaiton	2181	2177	2170	2168	2159
Missclassification	2	5	13	15	24
Success rate in %	99.9	99.8	99.4	99.3	98.9
Error rate in %	0.1	0.2	0.6	0.7	1.1
Total success rate in %	99.9	99.8	99.7	99.6	99.5
Total error rate in %	0.1	0.2	0.3	0.4	0.5

Table 23 - ANN, detecion of 10 byte message

Message 10 bytes	800 x 600	1024 x 768	1280 x 1024	1440 x 900	1680 x 1050
Samples total	2182	2183	2183	2183	2183
Correct classificcaiton	2180	2178	2170	2168	2159
Missclassification	2	5	13	15	24
Success rate in %	99.9	99.8	99.4	99.3	98.9
Error rate in %	0.1	0.2	0.6	0.7	1.1
Total success rate in %	99.9	99.9	99.7	99.6	99.5
Total error rate in %	0.1	0.1	0.3	0.4	0.5

Table 24 - ANN, detecion of 15 byte message

Message 15 bytes	800 x 600	1024 x 768	1280 x 1024	1440 x 900	1680 x 1050
Samples total	2183	2182	2183	2182	2183
Correct classificcaiton	2181	2177	2170	2167	2159
Missclassification	2	5	13	15	24
Success rate in %	99.9	99.8	99.4	99.3	98.9
Error rate in %	0.1	0.2	0.6	0.7	1.1
Total success rate in %	99.9	99.9	99.7	99.6	99.5
Total error rate in %	0.1	0.1	0.3	0.4	0.5

Table 25- ANN, detecion of 30 byte message

Message 30 bytes	800 x 600	1024 x 768	1280 x 1024	1440 x 900	1680 x 1050
Samples total	2183	2182	2183	2182	2183
Correct classificcaiton	2181	2177	2170	2168	2159
Missclassification	2	5	13	15	24
Success rate in %	99.9	99.8	99.4	99.3	98.9
Error rate in %	0.1	0.2	0.6	0.7	1.1
Total success rate in %	99.9	99.9	99.7	99.6	99.5
Total error rate in %	0.1	0.1	0.3	0.4	0.5

Table 26 - ANN, detecion of 75 byte message

Message 75 bytes	800 x 600	1024 x 768	1280 x 1024	1440 x 900	1680 x 1050
Samples total	2182	2183	2183	2182	2183
Correct classificcaiton	2180	2178	2170	2167	2159
Missclassification	2	5	13	15	24
Success rate in %	99.9	99.8	99.4	99.3	98.9
Error rate in %	0.1	0.2	0.6	0.7	1.1
Total success rate in %	99.9	99.9	99.7	99.6	99.5
Total error rate in %	0.1	0.1	0.3	0.4	0.5

Table 27 - ANN, detecion of 150 byte message

Message 150 bytes	800 x 600	1024 x 768	1280 x 1024	1440 x 900	1680 x 1050
Samples total	2183	2183	2183	2182	2183
Correct classificcaiton	2181	2178	2170	2167	2159
Missclassification	2	5	13	15	24
Success rate in %	99.9	99.8	99.4	99.3	98.9
Error rate in %	0.1	0.2	0.6	0.7	1.1
Total success rate in %	99.9	99.9	99.7	99.6	99.5
Total error rate in %	0.1	0.1	0.3	0.4	0.5

Table 28 - ANN, detecion of 300 byte message

Message 300 bytes	800 x 600	1024 x 768	1280 x 1024	1440 x 900	1680 x 1050
Samples total	2183	2183	2183	2182	2183
Correct classificcaiton	2181	2178	2170	2167	2159
Missclassification	2	5	13	15	24
Success rate in %	99.9	99.8	99.4	99.3	98.9
Error rate in %	0.1	0.2	0.6	0.7	1.1
Total success rate in %	99.9	99.9	99.7	99.6	99.5
Total error rate in %	0.1	0.1	0.3	0.4	0.5

Table 29 - ANN, detecion of 600 byte message

Message 600 bytes	800 x 600	1024 x 768	1280 x 1024	1440 x 900	1680 x 1050
Samples total	2183	2183	2183	2182	2183
Correct classificcaiton	2181	2178	2170	2167	2159
Missclassification	2	5	13	15	24
Success rate in %	99.9	99.9	99.4	99.3	98.9
Error rate in %	0.1	0.2	0.6	0.7	1.1
Total success rate in %	99.9	99.9	99.7	99.6	99.5
Total error rate in %	0.1	0.1	0.3	0.4	0.5

Table 30 – ANN, classification of cover images

COVER images	800 x 600	1024 x 768	1280 x 1024	1440 x 900	1680 x 1050	1920 x 1200	1920 x 1440	2560 x 1600
Samples total	2182	2183	2183	2183	2183	2183	2183	2183
Correct classificcaiton	2179	2182	2182	2182	2183	2183	2183	2182
Missclassification	3	1	1	1	0	0	0	1
Success rate in %	99.9	99.9	99.9	99.9	100	100	100	99.9
Error rate in %	0.1	0.1	0.1	0.1	0	0	0	0.1

The artificial neural networks reached better results than Stegdetect tool. The total error rate was maximum 0.5 % compared to Stegdetect tool where the total rate was minimal 2.2% and maximal 7.5%.

10 FINAL TEST

The final test was performed with different settings of neurons in hidden layer (1 to 20) and 9 combinations of transfer functions (logistic sigmoid, saturated linear and hyperbolic tangens for hidden layer and output neuron). The tests were carried out for each algorithm individually. The setting of number of hidden neurons and transfer functions is written in each table. Only the best results for each algorithm are presented here. Other suitable solutions are stated in Appendix A, B, C, D, which cover results with higher total error rate in comparison with results presented in this chapter.

10.1.1 Neural network topology

The neural network used for Outguess classification is feedforward network with sixty four neurons at the input, twelve neuron in one hidden layer and one neuron at the output. Transfer function was sigmoid and satured linear function at the output.

10.1.2 Results

Table 31 - Settings and overall statistic of Outguess detection

TOTAL percentage with OutGuess Hidden neurons: {12} functions: Sigmoid-Output_Saturated_Linear	
Cover total	5246
Cover errors	2
Cover % error	0.0381243
Cover % success	99.9619
Stego total	135 891
Stego errors	0
Stego % error	0.
Stego % success	100.
Total % error	0.00141706
Total % success	99.9986

Table 32 – Total error level of results for 5 to 30 byte long hidden message detection

% error with OutGuess Hidden neurons: {12} functions: Sigmoid-Output_Saturated_Linear															
	Cover			5 B			10 B			15 B			30 B		
	nr. of err.	nr. of sampl.	err. in %	nr. of err.	nr. of sampl.	err. in %	nr. of err.	nr. of sampl.	err. in %	nr. of err.	nr. of sampl.	err. in %	nr. of err.	nr. of sampl.	err. in %
original	0	582	0.	0	1970	0.	0	1961	0.	0	1956	0.	0	1919	0.
800x600	2	583	0.343053	0	1982	0.	0	1982	0.	0	1982	0.	0	1982	0.
1024x768	0	583	0.	0	1981	0.	0	1980	0.	0	1979	0.	0	1975	0.
1280x1024	0	583	0.	0	1979	0.	0	1979	0.	0	1978	0.	0	1975	0.
1440x900	0	583	0.	0	1981	0.	0	1980	0.	0	1980	0.	0	1972	0.
1680x1050	0	583	0.	0	1978	0.	0	1975	0.	0	1974	0.	0	1965	0.
1920x1200	0	583	0.	0	1976	0.	0	1973	0.	0	1973	0.	0	1966	0.
1920x1440	0	583	0.	0	1975	0.	0	1969	0.	0	1968	0.	0	1949	0.
2560x1600	0	583	0.	0	1973	0.	0	1971	0.	0	1970	0.	0	1956	0.

Table 33 - Total error level of results for 75 to 300 byte long hidden message detection

% error with OutGuess Hidden neurons: {12} functions: Sigmoid-Output_Saturated_Linear												
	75 B			150 B			300 B			600 B		
	nr. of err.	nr. of sampl.	err. in %	nr. of err.	nr. of sampl.	err. in %	nr. of err.	nr. of sampl.	err. in %	nr. of err.	nr. of sampl.	err. in %
original	0	1827	0.	0	1650	0.	0	1297	0.	0	1011	0.
800x600	0	1981	0.	0	1981	0.	0	1976	0.	0	1951	0.
1024x768	0	1969	0.	0	1958	0.	0	1940	0.	0	1897	0.
1280x1024	0	1958	0.	0	1927	0.	0	1870	0.	0	1786	0.
1440x900	0	1960	0.	0	1936	0.	0	1880	0.	0	1795	0.
1680x1050	0	1938	0.	0	1907	0.	0	1823	0.	0	1669	0.
1920x1200	0	1940	0.	0	1897	0.	0	1791	0.	0	1595	0.
1920x1440	0	1909	0.	0	1850	0.	0	1737	0.	0	1510	0.
2560x1600	0	1917	0.	0	1840	0.	0	1667	0.	0	1367	0.

10.2 Steghide

10.2.1 Neural network topology

The neural network used for Steghide classification is feedforward network with sixty four neurons at the input, one neuron in one hidden layer and one neuron at the output. Transfer function was sigmoid and satured linear function at the output.

10.2.2 Results

Table 34 - Settings and overall statistic of Steghide detection

TOTAL percentage with Steghide Hidden neurons: {1} functions: Sigmoid-Output_Saturated_Linear	
Cover total	5246
Cover errors	32
Cover % error	0.609989
Cover % success	99.39
Stego total	142 772
Stego errors	3248
Stego % error	2.27496
Stego % success	97.725
Total % error	2.21595
Total % success	97.7841

Table 35 – Total error level of results for 5 to 30 byte long hidden message detection

% error with Steghide Hidden neurons: {1} functions: Sigmoid-Output_Saturated_Linear															
	Cover			5 B			10 B			15 B			30 B		
	nr. of err.	nr. of sampl.	err. in %	nr. of err.	nr. of sampl.	err. in %	nr. of err.	nr. of sampl.	err. in %	nr. of err.	nr. of sampl.	err. in %	nr. of err.	nr. of sampl.	err. in %
original	23	582	3.95189	35	1983	1.765	35	1983	1.765	35	1983	1.765	35	1983	1.765
800x600	5	583	0.857633	0	1983	0.	0	1983	0.	0	1983	0.	0	1983	0.
1024x768	3	583	0.51458	0	1983	0.	0	1983	0.	0	1983	0.	0	1983	0.
1280x1024	0	583	0.	0	1983	0.	0	1983	0.	0	1983	0.	0	1983	0.
1440x900	1	583	0.171527	0	1983	0.	0	1983	0.	0	1983	0.	0	1983	0.
1680x1050	0	583	0.	0	1983	0.	0	1983	0.	0	1983	0.	0	1983	0.
1920x1200	0	583	0.	0	1983	0.	0	1983	0.	0	1983	0.	0	1983	0.
1920x1440	0	583	0.	9	1983	0.453858	9	1983	0.453858	9	1983	0.453858	9	1983	0.453858
2560x1600	0	583	0.	363	1983	18.3056	364	1983	18.356	365	1983	18.4065	362	1983	18.2552

Table 36 - Total error level of results for 75 to 300 byte long hidden message detection

% error with Steghide Hidden neurons: {1} functions: Sigmoid-Output_Saturated_Linear												
	75 B			150 B			300 B			600 B		
	nr. of err.	nr. of sampl.	err. in %	nr. of err.	nr. of sampl.	err. in %	nr. of err.	nr. of sampl.	err. in %	nr. of err.	nr. of sampl.	err. in %
original	35	1983	1.765	35	1983	1.765	35	1983	1.765	36	1983	1.81543
800x600	0	1983	0.	0	1983	0.	0	1982	0.	0	1982	0.
1024x768	0	1983	0.	0	1983	0.	0	1983	0.	0	1982	0.
1280x1024	0	1983	0.	0	1983	0.	0	1983	0.	0	1983	0.
1440x900	0	1983	0.	0	1983	0.	0	1983	0.	0	1983	0.
1680x1050	0	1983	0.	0	1982	0.	0	1983	0.	0	1983	0.
1920x1200	0	1983	0.	0	1983	0.	0	1983	0.	0	1983	0.
1920x1440	9	1983	0.453858	9	1983	0.453858	9	1983	0.453858	9	1983	0.453858
2560x1600	365	1983	18.4065	360	1983	18.1543	360	1983	18.1543	356	1983	17.9526

10.3 F5 algorithm

10.3.1 Neural network topology

The neural network used for F5 classification is feedforward network with sixty four neurons at the input, fifteen neuron in one hidden layer and one neuron at the output. Transfer function was satured linear and hyperbolic tangent function at the output.

10.3.2 Results

Table 37 - Settings and overall statistic of F5 detection

TOTAL percentage with algorithm F5 Hidden neurons: {15} functions: Saturated_Linear-Output_HyperbolicTangent	
Cover total	9746
Cover errors	8
Cover % error	0.082085
Cover % success	99.9179
Stego total	77 314
Stego errors	24
Stego % error	0.0310422
Stego % success	99.969
Total % error	0.0367563
Total % success	99.9632

Table 38 – Total error level of results for 5 to 30 byte long hidden message detection

% error with algorithm F5 Hidden neurons: {15} functions: Saturated_Linear-Output_HyperbolicTangent															
	Cover			5 B			10 B			15 B			30 B		
	nr. of err.	nr. of sampl.	err. in %	nr. of err.	nr. of sampl.	err. in %	nr. of err.	nr. of sampl.	err. in %	nr. of err.	nr. of sampl.	err. in %	nr. of err.	nr. of sampl.	err. in %
original	3	1082	0.277264												
800x600	3	1083	0.277008	0	1933	0.	0	1932	0.	0	1933	0.	0	1933	0.
1024x768	2	1083	0.184672	0	1932	0.	0	1933	0.	0	1932	0.	0	1932	0.
1280x1024	0	1083	0.	0	1933	0.	0	1933	0.	0	1933	0.	0	1933	0.
1440x900	0	1083	0.	2	1933	0.103466	2	1933	0.103466	2	1932	0.10352	2	1933	0.103466
1680x1050	0	1083	0.	1	1933	0.0517331	1	1933	0.0517331	1	1933	0.0517331	1	1933	0.0517331
1920x1200	0	1083	0.												
1920x1440	0	1083	0.												
2560x1600	0	1083	0.												

Table 39 - Total error level of results for 75 to 300 byte long hidden message detection

% error with algorithm F5 Hidden neurons: {15} functions: Saturated_Linear-Output_HyperbolicTangent												
	75 B			150 B			300 B			600 B		
	nr. of err.	nr. of sampl.	err. in %	nr. of err.	nr. of sampl.	err. in %	nr. of err.	nr. of sampl.	err. in %	nr. of err.	nr. of sampl.	err. in %
original												
800x600	0	1932	0.	0	1933	0.	0	1933	0.	0	1933	0.
1024x768	0	1933	0.	0	1933	0.	0	1933	0.	0	1933	0.
1280x1024	0	1933	0.	0	1933	0.	0	1933	0.	0	1933	0.
1440x900	2	1933	0.103466	2	1933	0.103466	2	1933	0.103466	2	1933	0.103466
1680x1050	1	1933	0.0517331	1	1933	0.0517331	1	1933	0.0517331	1	1933	0.0517331
1920x1200												
1920x1440												
2560x1600												

10.4 PQ algorithm

10.4.1 Neural network topology

The neural network used for PQ algorithm classification is feedforward network with sixty four neurons at the input, one neuron in one hidden layer and one neuron at the output. Transfer function was sigmoid and satured linear function at the output.

10.4.2 Results

Table 40 - Settings and overall statistic of PQ detection

TOTAL percentage with PQ Hidden neurons: {1} functions: Sigmoid-Output_Saturated_Linear	
Cover total	22 711
Cover errors	1
Cover % error	0.00440315
Cover % success	99.9956
Stego total	126 599
Stego errors	773
Stego % error	0.610589
Stego % success	99.3894
Total % error	0.518385
Total % success	99.4816

Table 41 – Total error level of results for 5 to 30 byte long hidden message detection

% error with PQ
Hidden neurons: {1}
functions: Sigmoid-Output_Saturated_Linear

	Cover			5 B			10 B			15 B			30 B		
	nr. of err.	nr. of sampl.	err. in %	nr. of err.	nr. of sampl.	err. in %	nr. of err.	nr. of sampl.	err. in %	nr. of err.	nr. of sampl.	err. in %	nr. of err.	nr. of sampl.	err. in %
original	0	583	0.	37	1977	1.87152	37	1979	1.86963	39	1980	1.9697	37	1980	1.86869
800x600	0	2766	0.	34	1978	1.71891	33	1974	1.67173	32	1980	1.61616	32	1980	1.61616
1024x768	0	4949	0.	13	1977	0.657562	12	1974	0.607903	12	1978	0.606673	11	1980	0.555556
1280x1024	0	2766	0.	4	1977	0.202327	4	1976	0.202429	4	1978	0.202224	4	1979	0.202122
1440x900	0	2766	0.	2	1980	0.10101	2	1975	0.101266	2	1979	0.101061	2	1979	0.101061
1680x1050	1	583	0.171527	1	1978	0.0505561	1	1979	0.0505306	1	1979	0.0505306	1	1979	0.0505306
1920x1200	0	2766	0.												
1920x1440	0	2766	0.	5	1979	0.252653	5	1979	0.252653	5	1979	0.252653	4	1979	0.202122
2560x1600	0	2766	0.	4	1975	0.202532	4	1977	0.202327	4	1978	0.202224	5	1977	0.252908

Table 42 - Total error level of results for 75 to 300 byte long hidden message detection

% error with PQ
Hidden neurons: {1}
functions: Sigmoid-Output_Saturated_Linear

	75 B			150 B			300 B			600 B		
	nr. of err.	nr. of sampl.	err. in %	nr. of err.	nr. of sampl.	err. in %	nr. of err.	nr. of sampl.	err. in %	nr. of err.	nr. of sampl.	err. in %
original	38	1980	1.91919	36	1980	1.81818	37	1979	1.86963	37	1979	1.86963
800x600	33	1979	1.66751	33	1979	1.66751	31	1979	1.56645	31	1978	1.56724
1024x768	11	1980	0.555556	11	1980	0.555556	10	1979	0.505306	11	1979	0.555836
1280x1024	4	1979	0.202122	4	1978	0.202224	4	1978	0.202224	4	1978	0.202224
1440x900	2	1979	0.101061	2	1979	0.101061	2	1979	0.101061	2	1978	0.101112
1680x1050	1	1979	0.0505306	1	1979	0.0505306	1	1978	0.0505561	1	1976	0.0506073
1920x1200												
1920x1440	4	1979	0.202122	4	1979	0.202122	4	1978	0.202224	4	1976	0.202429
2560x1600	5	1977	0.252908	4	1976	0.202429	4	1973	0.202737	4	1972	0.20284

11 APPLICABILITY OF THE SOLUTION RESULTS

During the preprocessing of sample images for this dissertation project a new image sampling method for steganalysis have been found. The results positively prooved the draft of image sampling by the Huffman coding for the steganalysis by means of neural networks.

Sampling results analysis represented by Huffman coding histogram as showed in Fig. 12. and Fig. 13 has been proved as fully functional, however the complexity of full Huffman coding histogram has strong impact on learning efficiency of Levenberg–Marquardt algorithm. This was a reason for comprehensive analysis of available typologies of neural networks. Main goal was to find balanced learning efficiency and classification accuracy, those findings are shown in chapter 10.

There are another analysis aimed on improvement of learning process by datamining, this analysis have been caried out as part of dissertation of Ing. Michal Prochazka. Analysis have been focused on reducing input vector of training sets for neural networks. Results have been published in article *Datamining Optimization of Steganalysis by means of Neural Network* Competitiveness and Organizational Security conference at UTB 2010.

Results of dissertation and research will be used for proof of concept application for steganalysis. There are several private objects with interest in steganalytical application. Results and programs will be used for forensic analysis with focus on collecting evidence of covert communication.

12 CONCLUSION

This thesis deals with a detection of a steganography content in images inserted by four programmes – OutGuess, Steghide, CipherAWT (F5 algorithm) and PQ. The detection is done by means of the feedforward artificial neural network with one hidden layer. ANN was successful in nearly 100% of classification cases shown in chapter 10. The root mean square error in the training phase was almost zero which says that training was successful as well. This approach shows a promising way of detecting hidden content in the images to avoid taking secret information out of the company. The aim for the future is to develop a steganalysis detector, The detector might be part of the outgoing post servers. This will not decode the messages itself, this research is focused only on recognizing the use of the steganography tools (encoder) on the analyzed JPEG pictures.

Further work could focus on the identification with a particular embedding mechanism and classifies the steganography application. This research could also be extended for future analysis of the still image steganography and video steganalysis. The sampling methodology mentioned in this thesis could be used for staticical analysis of the TCP/IP packet headers.

Training data sets obtained from the JPEG samples could be used for future research for benchmarking different methods of teaching artificial neural networks. When the research had been carried out by several tests aimed on datamaning over the trainig data set for estimating balanced learning efficiency (speed of learning process) and classification accuracy. Future analysis could be aimed on self arranged network typology by methods of symbolic regression like Genetic Programming, Grammatical Evolution, Analytic Programming and others, i.e. superstructure of evolutionary optimization algorithms.

13 THE LIST OF AUTOR'S PUBLICATION ACTIVITIES

1. Oplatková, Z., Hološka, J., Zelinka, I., Šenkeřík, R.: Detection of Steganography Content Inserted by Steghide by means of Neural Networks, Vysoké učení technické v Brně, Fakulta strojního inženýrství, MENDEL 2008 14th International Coference on Soft Computing, Brno, 2008, 166-171, ISBN 978-80-214-3675-6

2. Oplatková, Z., Hološka, J., Zelinka, I., Šenkeřík, R.: Steganography Detection by means of Neural Networks, IEEE Operations Center, Nineteenth International Workshop on Database and Expert Systems Applications, Piscataway, 2008, 571-576, ISBN 978-0-7695-3299-8

3. Hološka, J., Oplatková, Z., Zelinka, I., Šenkeřík, R.: Detection of Stegoimages by F5 Programme by means of Neural Networks, Vysoké učení technické v Brně, Fakulta strojního inženýrství, MENDEL 2009 15th International Coference on Soft Computing, Brno, 2009, 186-191, ISBN 978-80-214-3884-2

4. Oplatková, Z., Hološka, J., Zelinka, I., Šenkeřík, R.: Steganography Detection by means of Neural Networks, XI Annual Intenrational Conference: Internet, Competitiveness and Organisational Security in Knowledge Society, Zlín, 2009, ISBN 978-80-7318-828-3

5. Oplatková, Z., Hološka, J., Zelinka, I., Šenkeřík, R.: Detection of Steganography Inserted by OutGuess and Steghide by means of Neural Networks, AMS2009 Asia Modelling Symposium 2009, IEEE Computer Society, Piscataway, 2009, ISBN 978-0-7695-3648-4

6. Macurová, L., Hološka, J. Význam lidského faktoru v oblasti managementu informační bezpečnosti. Personálny Manažment - Trendy na trhu práce v kontexte hospodárskej krízy. 1. vyd. Trenčín : [s.n.], 2009. s. 183-189. ISBN 9788080754037.

7. Hološka, J., Oplatková, Z., Zelinka, I. Steganografie jako hrozba úniku kritických obchodních informací a její detekce pomocí umělých neuronových sítí. TRILOBIT [online]. 2009 ISSN 1804-1795.

8. Hološka, J., Oplatková, Z., Zelinka, I. Steganografie jako hrozba úniku kritických obchodních informací a její detekce pomocí umělých neuronových sítí. Security Magazín, 2010, ISSN: 1210-8723

9. Oplatkova Z., Senkerik R., Zelinka I. Holoska J.: Synthesis of Control Law for Chaotic Logistic Equation - Preliminary study, AMS 2010, Kota Kinabalu, Borneo, Malaysia, IEEE, 6 p., ISBN 978-0-7695-4062-7

10. Oplatkova Z., Senkerik R., Zelinka I. Holoska J.: Synthesis of Control Law for Chaotic Henon System - Preliminary study, ECMS 2010, Kuala Lumpur, Malaysia, p. 277-282, ISBN 978-0-9564944-0-5

11. Prochazka M., Oplatkova Z., Holoska J.: Datamining Optimization of Steganalysis by means of Neural Network, in Internet, Competitiveness and Organizational Security 2010, UTB Zlín, Czech Republic, ISBN 978 - 83 - 61645 - 16 – 0

12. Holoska J., Oplatkova Z., Senkerik R., Zelinka I.: Comparison Between Neural Network Steganalysis and Linear Classification Method Stegdetect, CIMSim 2010, Bali, Indonesie, IEEE, ISBN: 978-0-7695-4262-1

13. Oplatkova Z., Holoska J., Zelinka I., Senkerik R., Jasek R.: Steganalysis of PQ Algorithm By Means Of Neural Networks, ECMS 2011, Krakow, Poland, p. 446 - 451, ISBN: 978-0-9564944-2-9

14. Prochazka M., Oplatkova Z., Holoska J., Gerlich V.: Optimization Of Neural Network Inputs By Feature Selection Methods, ECMS 2011, Krakow, Poland, p.440-445, ISBN: 978-0-9564944-2-9

14 REFERENCES

[1] Cole, E., Krutz D. R. Hiding Sight.. United States of America : Wiley Publishing, Inc., 2003. 321 s. ISBN 0-471-44449-9.

[2] Goldwasser, S. , Bellare, M. . Lecture Notes on Cryptography. Cambridge, Massachusetts [MIT] : [s.n.], 2001. 283 s.

[3] Hertz J., Kogh A. and Palmer R. G. Introduction to the Theory of Neural Computation, Addison – Wesley 1991

[4] http://steghide.sourceforge.net/

[5] http://linux01.gwdg.de/~alatham/stego.html

[6] http://www.outguess.org/detection.php

[7] http://www.outguess.org/

[8] Provos, N. , Honeyman, P. Detecting Steganographic Content on the Internet. CITI Technical Report 01-11, 2001

[9] Provos, N. , Honeyman, P. Hide and Seek: An Introduction to Steganography. IEEE Security & Privacy 1(3): 32-44 (2003)

[10] Westfeld, A. , Pfitzmann, A. . Attacks on Steganographic Systems. In Proceedings of Information Hiding - Third International Workshop. Springer Verlag, September 1999.

[11] Fridrich, J., Goljan, M., and Hogea, D. "Steganalysis of JPEG Images: Breaking the F5 Algorithm." 5th Information Hiding Workshop, Noordwijkerhout, The Netherlands, Oct. 2002. URL: http://www.ws.binghamton.edu/fridrich/Research/f5.pdf. Last accessed: 2003-12-24.

[12] Fridrich, J., Goljan, M., and Hogea, D.: New Methodology for Breaking Steganographic Techniques for JPEGs. Submitted to SPIE: Electronic Imaging 2003, Security and Watermarking of Multimedia Contents. Santa Clara, California, 2003

[13] Fridrich, J. Feature-Based Steganalysis for JPEG Images and Its Implications for Future Design of Steganographic Schemes. In Proceedings of Information Hiding' 2004. pp.67~81

[14] Farid, H. and Siwei Lyu. Detecting Hidden Messages Using Higher-Order Statistics and Support Vector Machines. 5th Information Hiding Workshop, Noordwijkerhout, Netherlands, Oct. 7−9, 2002

[15] Hempstalk, K. Hiding Behind Corners: Using Edges in Images for Better Steganography, Proceedings of the Computing Women's Congress, Hamilton, New Zealand, 11- 19 February 2006.

[16] http://www.cs.waikato.ac.nz/ml/weka/

[17] Gü l, G.; Dirik, A. E. ; Avcibas ,I. . Steganalytic Features for JPEG Compression-Based Perturbed Quantization. IEEE SIGNAL PROCESSING LETTERS. MARCH 2007, 3, s.

[18] Singh, S .. The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography. New York : Doubleday of New York, 1999. 432 s. ISBN 0-385-49531-5.

[19] http://www.jjtc.com/Steganography/tools.html

[20] http://www.outguess.org/

[21] Steganography: Code Recognition Technology. [online]. FUJITSU LABORATORIES, March, 2008 [cit. 2010-12-1]. URL: <http://jp.fujitsu.com/group/labs/downloads/en/business/activities/activities-4/fujitsu-labs-imagevoice-003-en.pdf>.

[22] Burris, D.. [online]. 2009 [cit. 2011-08-07]. Steganography Concepts. Dostupné z WWW: <http://df.shsu.edu/crypto/Steganography2/Steganographyhtml.php>.

[23] Khairullah, M. A.. Novel Text Steganography System in Cricket Match Scorecard. International Journal of Computer Applications [online]. May 2011, 21, [cit. 2011-08-07]. Dostupný z WWW: <http://www.ijcaonline.org/volume21/number9/pxc3873462.pdf>.

[24] MessageLabs Intelligence. 2010 Annual Security Report. [online]. 2010, [cit. 2010-11-08]. Dostupný z WWW: <http://www.symanteccloud.com/mlireport/MessageLabsIntelligence_2010_Annual_Report_FINAL.pdf>.

[25] Frequency analysis. In Wikipedia : the free encyclopedia [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2011-01-06]. WWW: <http://en.wikipedia.org/wiki/Frequency_analysis>.

[26] Wayner, P. Disappearing Cryptography, Second Edition: Information Hiding: Steganography & Watermarking (The Morgan Kaufmann Series in Software Engineering and Programming)., Morgan Kaufmann; 2 edition, 2002. 413 s. ISBN 978-1558607699.

[27] Katzenbeisser, S; Petitcolas, F. Information Hiding Techniques for Steganography and Digital Watermarking. [s.l.] : Artech House Print on Demand, 1999. 220 s. ISBN 978-1580530354.

[28] Noto, M. MP3Stego: Hiding Text in MP3 File, September 15, 2001 <http://www.sans.org/reading_room/whitepapers/stenganography/mp3stego-hiding-text-mp3-files_550>

[29] Shankaranarayanan, , M. Audio file steganography. [online]. 0000, [cit. 2011-01-06]. WWW: <http://www.cise.ufl.edu/~smanamal/steganography.htm>.

[30] Chen, Wenxiao , Jing, Cai, Siwei, Li. Advanced Parallel Processing Technologies. [s.l.] : Springer Berlin / Heidelberg, 2007. Dostupný z WWW: <http://www.springerlink.com/content/17915u4814u28131/>. ISBN 978-3-540-76. An Anti-statistical Analysis LSB Steganography Incorporating Extended Cat-Mapping, s. 468-476.

[31] Kessler, Gary C. . Steganography: Hiding Data Within Data. [online]. 2001, [cit. 2009-04-29]. Dostupný z WWW: <http://www.garykessler.net/library/steganography.html>.

[32] Software OutGuess, www.outguess.org

[33] Defending Against Statistical Steganalysis Niels Provos, 10th USENIX Security Symposium. Washington, DC, August 2001

[34] Hetzl S.: Steghide (1) - Linux man page [online]. [cit. 2008-05-21]. available from WWW: <http://steghide.sourceforge.net/documentation/manpage.php>.

[35] Westfeld, A.: High Capacity Despite Better Steganalysis (F5 – –A Steganographic Algorithm). In: Moskowitz, I.S. (eds.): Information Hiding. 4th International Workshop. Lecture Notes in Computer Science, Vol.2137. Springer-Verlag, Berlin Heidelberg New York (2001) 289 – – 302

[36] Johnson, S. Introduction to JPEG [online]. 1999 [cit. 2009-03-18]. Dostupný z WWW: <http://www.faqs.org/faqs/compression-faq/part2/section- 6.html>.

[37] Farid, H., Detecting Hidden Messages Using Higher- Order Statistical Models , International Conference on Image Processing, Rochester, NY, 2002

[38] Provos, N. Steganography Detection with Stegdetect [online]. 1999 , September 5 2004 [cit. 2009-03-18]. Dostupný z WWW: <http://www.outguess.org/detection.php>.

[39] Benes, M. Komprese: Huffmanovo kódování, cited 2008-02-15, http://www.cs.vsb.cz/benes/vyuka/pte/texty/komprese/ch02s02.html

[40] McCulloch W. S., Pitts W. A logical calculus of ideas immanent in nervous activity, In Bulletin of Mathematical Biophysics, 5:115, 133, 1943

[41] Reitermanova Z. Feedforward Neural Networks – Architecture Optimization and Knowledge Extraction, WDS'08 Proceedings of Contributed Papers, Part I, 159–164, 2008, MATFYZPRESS, ISBN 978-80-7378-065-4

[42] Lampinen J., Zelinka I., New Ideas in Optimization – Mechanical Engineering Design Optimization by Differential Evolution. Volume 1. London: McGraw-Hill, 1999. 20 p. ISBN 007-709506-5

[43] Zelinka I. 2004a, „SOMA – Self Organizing Migrating Algorithm“, In: B.V. Babu, G. Onwubolu (eds),), New Optimization Techniques in Engineering Springer-Verlag, 2004, ISBN 3-540-20167X

[44] Zelinka I., Oplatková Z., Šeda M., Ošmera P., Včelař F., Evoluční výpočetní techniky - principy a aplikace, BEN, Praha, 2008, 550 s., ISBN 80-7300-218-3

[45] JPEGSnoop - http://www.impulseadventure.com/photo/jpeg-snoop.html

[46] ImageMagick - http://www.imagemagick.org/script/index.php

[47] Sedlak, J.: Digital Image Information Processing for the Steganalysis Using Neural Networks, Master thesis, Tomas Bata University in Zlin, Czech edition, Zlin, Czech Republic, 63 p., 2010.

[48] Holoska J., Odhalování steganografie pomocí neuronových sítí, diploma thesis in Czech edition, UTB Zlín, 2008

[49] Hološka, J., Oplatková, Z., Zelinka, I.: Steganografie jako hrozba úniku kritických obchodních informací a její detekce pomocí umělých neuronových sítí. Czceh edi., TRILOBIT [online]. 2009 ISSN 1804-1795

15 CITED BY

Oplatková, Z., Hološka, J., Zelinka, I., Šenkeřík, R.: Steganography Detection by means of Neural Networks, IEEE Operations Center, Nineteenth International Workshop on Database and Expert Systems Applications, Piscataway, 2008, 571-576, ISBN-ISSN 978-0-7695-3299-8

cited by:

Mansor S., Din R., Samsudin A.: Analysis of natural language steganography, International Journal of Computer Science and Security, vol. 2, issue 3., 2009

Cerkez P.S., Cannady J.D.: Automated detection of semagram-laden images using adaptive neural networks, Proceedings of SPIE-The International Society for Optical Engineering, 2010, ISBN: 978-0-8194-8172-6

Veena H. B., Krishna S., Deepa S., Venugopal K.R., Patnaik L.M.: JPEG steganalysis usign HBCL statistics and FR index, Lecture Notes in Computer Science, 2010, Volume 6122/2010, 105-112, DOI: 10.1007/978-3-642-13601-6_13

Oplatková, Z., Hološka, J., Zelinka, I., Šenkeřík, R. : Detection of Steganography Inserted by OutGuess and Steghide by means of Neural Networks, AMS2009 Asia Modelling Symposium 2009, IEEE Computer Society, Piscataway, 2009, ISBN 978-0-7695-3648-4

cited by:

FORD, Blake W. . File Format Extension Through Steganography. San Marcos,Texas, 2010. 164 s. Master's thesis. Texas State University-San Marcos, Dept. of Computer Science. Dostupné z WWW: <http://ecommons.txstate.edu/cscitad/7/>.

APPENDIXES

Apendixes A, B, C, D contain tables of results with higher total error rate in comparison with results in chapter 10. Apendixes represent results of different settings of artifical neural networks presented in this thesis. Apendix A covers detection of Outguess algorithm, Apendix B the F5 algorithm, Apendix C the Steghide algorithm and Apendix D the PQ algorithm.

Appendix A: Successful results for OutGuess algorithm

TOTAL percentage with OutGuess Hidden neurons: {1} functions: Hyperbolic_Tangent-Output_Saturated_Linear	
Cover total	5246
Cover errors	46
Cover % error	0.876859
Cover % success	99.1231
Stego total	135 891
Stego errors	0
Stego % error	0.
Stego % success	100.
Total % error	0.0325924
Total % success	99.9674

TOTAL percentage with OutGuess Hidden neurons: {1} functions: Saturated_Linear-Output_HyperbolicTangent	
Cover total	5246
Cover errors	65
Cover % error	1.23904
Cover % success	98.761
Stego total	135 891
Stego errors	31
Stego % error	0.0228124
Stego % success	99.9772
Total % error	0.068019
Total % success	99.932

TOTAL percentage with OutGuess Hidden neurons: {1} functions: Sigmoid-Output_Hyperbolic_Tangent	
Cover total	5246
Cover errors	25
Cover % error	0.476554
Cover % success	99.5234
Stego total	135 891
Stego errors	0
Stego % error	0.
Stego % success	100.
Total % error	0.0177133
Total % success	99.9823

TOTAL percentage with OutGuess Hidden neurons: {4} functions: Hyperbolic_Tangent-Output_Hyperbolic_Tangent	
Cover total	5246
Cover errors	43
Cover % error	0.819672
Cover % success	99.1803
Stego total	135 891
Stego errors	16
Stego % error	0.0117741
Stego % success	99.9882
Total % error	0.0418034
Total % success	99.9582

TOTAL percentage with OutGuess Hidden neurons: {5} functions: Sigmoid-Output_Saturated_Linear	
Cover total	5246
Cover errors	3
Cover % error	0.0571864
Cover % success	99.9428
Stego total	135 891
Stego errors	56
Stego % error	0.0412095
Stego % success	99.9588
Total % error	0.0418034
Total % success	99.9582

TOTAL percentage with OutGuess Hidden neurons: {6} functions: Hyperbolic_Tangent-Output_Saturated_Linear	
Cover total	5246
Cover errors	0
Cover % error	0.
Cover % success	100.
Stego total	135 891
Stego errors	134
Stego % error	0.0986084
Stego % success	99.9014
Total % error	0.0949432
Total % success	99.9051

TOTAL percentage with OutGuess Hidden neurons: {6} functions: Saturated_Linear-Output_Saturated_Linear	
Cover total	5246
Cover errors	3
Cover % error	0.0571864
Cover % success	99.9428
Stego total	135 891
Stego errors	75
Stego % error	0.0551913
Stego % success	99.9448
Total % error	0.0552655
Total % success	99.9447

TOTAL percentage with OutGuess Hidden neurons: {6} functions: Sigmoid-Output_Saturated_Linear	
Cover total	5246
Cover errors	64
Cover % error	1.21998
Cover % success	98.78
Stego total	135 891
Stego errors	29
Stego % error	0.0213406
Stego % success	99.9787
Total % error	0.0658934
Total % success	99.9341

TOTAL percentage with OutGuess Hidden neurons: {6} functions: Sigmoid-Output_Sigmoid	
Cover total	5246
Cover errors	107
Cover % error	2.03965
Cover % success	97.9604
Stego total	135 891
Stego errors	16
Stego % error	0.0117741
Stego % success	99.9882
Total % error	0.0871494
Total % success	99.9129

TOTAL percentage with OutGuess Hidden neurons: {7} functions: Hyperbolic_Tangent-Output_Saturated_Linear	
Cover total	5246
Cover errors	7
Cover % error	0.133435
Cover % success	99.8666
Stego total	135 891
Stego errors	130
Stego % error	0.0956649
Stego % success	99.9043
Total % error	0.0970688
Total % success	99.9029

TOTAL percentage with OutGuess Hidden neurons: {7} functions: Sigmoid-Output_Hyperbolic_Tangent	
Cover total	5246
Cover errors	22
Cover % error	0.419367
Cover % success	99.5806
Stego total	135 891
Stego errors	58
Stego % error	0.0426813
Stego % success	99.9573
Total % error	0.0566825
Total % success	99.9433

TOTAL percentage with OutGuess Hidden neurons: {7} functions: Sigmoid-Output_Sigmoid	
Cover total	5246
Cover errors	105
Cover % error	2.00152
Cover % success	97.9985
Stego total	135 891
Stego errors	27
Stego % error	0.0198689
Stego % success	99.9801
Total % error	0.0935261
Total % success	99.9065

TOTAL percentage with OutGuess Hidden neurons: {9} functions: Hyperbolic_Tangent-Output_Saturated_Linear	
Cover total	5246
Cover errors	1
Cover % error	0.0190621
Cover % success	99.9809
Stego total	135 891
Stego errors	121
Stego % error	0.089042
Stego % success	99.911
Total % error	0.0864408
Total % success	99.9136

TOTAL percentage with OutGuess Hidden neurons: {9} functions: Saturated_Linear-Output_Saturated_Linear	
Cover total	5246
Cover errors	13
Cover % error	0.247808
Cover % success	99.7522
Stego total	135 891
Stego errors	32
Stego % error	0.0235483
Stego % success	99.9765
Total % error	0.0318839
Total % success	99.9681

TOTAL percentage with OutGuess Hidden neurons: {9} functions: Sigmoid-Output_Hyperbolic_Tangent	
Cover total	5246
Cover errors	36
Cover % error	0.686237
Cover % success	99.3138
Stego total	135 891
Stego errors	11
Stego % error	0.00809472
Stego % success	99.9919
Total % error	0.033301
Total % success	99.9667

TOTAL percentage with OutGuess Hidden neurons: {10} functions: Saturated_Linear-Output_Saturated_Linear	
Cover total	5246
Cover errors	14
Cover % error	0.26687
Cover % success	99.7331
Stego total	135 891
Stego errors	68
Stego % error	0.0500401
Stego % success	99.95
Total % error	0.0580996
Total % success	99.9419

TOTAL percentage with OutGuess Hidden neurons: {10} functions: Sigmoid-Output_Hyperbolic_Tangent	
Cover total	5246
Cover errors	15
Cover % error	0.285932
Cover % success	99.7141
Stego total	135 891
Stego errors	102
Stego % error	0.0750602
Stego % success	99.9249
Total % error	0.0828982
Total % success	99.9171

TOTAL percentage with OutGuess Hidden neurons: {12} functions: Saturated_Linear-Output_Sigmoid	
Cover total	5246
Cover errors	13
Cover % error	0.247808
Cover % success	99.7522
Stego total	135 891
Stego errors	122
Stego % error	0.0897778
Stego % success	99.9102
Total % error	0.0956517
Total % success	99.9043

TOTAL percentage with OutGuess Hidden neurons: {12} functions: Sigmoid-Output_Hyperbolic_Tangent	
Cover total	5246
Cover errors	32
Cover % error	0.609989
Cover % success	99.39
Stego total	135 891
Stego errors	74
Stego % error	0.0544554
Stego % success	99.9455
Total % error	0.0751043
Total % success	99.9249

TOTAL percentage with OutGuess Hidden neurons: {12} functions: Sigmoid-Output_Saturated_Linear	
Cover total	5246
Cover errors	2
Cover % error	0.0381243
Cover % success	99.9619
Stego total	135 891
Stego errors	0
Stego % error	0.
Stego % success	100.
Total % error	0.00141706
Total % success	99.9986

TOTAL percentage with OutGuess Hidden neurons: {13} functions: Sigmoid-Output_Saturated_Linear	
Cover total	5246
Cover errors	7
Cover % error	0.133435
Cover % success	99.8666
Stego total	135 891
Stego errors	8
Stego % error	0.00588707
Stego % success	99.9941
Total % error	0.010628
Total % success	99.9894

TOTAL percentage with OutGuess Hidden neurons: {14} functions: Saturated_Linear-Output_HyperbolicTangent	
Cover total	5246
Cover errors	9
Cover % error	0.171559
Cover % success	99.8284
Stego total	135 891
Stego errors	108
Stego % error	0.0794755
Stego % success	99.9205
Total % error	0.0828982
Total % success	99.9171

TOTAL percentage with OutGuess Hidden neurons: {14} functions: Sigmoid-Output_Sigmoid	
Cover total	5246
Cover errors	54
Cover % error	1.02936
Cover % success	98.9706
Stego total	135 891
Stego errors	26
Stego % error	0.019133
Stego % success	99.9809
Total % error	0.0566825
Total % success	99.9433

TOTAL percentage with OutGuess Hidden neurons: {15} functions: Saturated_Linear-Output_Sigmoid	
Cover total	5246
Cover errors	71
Cover % error	1.35341
Cover % success	98.6466
Stego total	135 891
Stego errors	0
Stego % error	0.
Stego % success	100.
Total % error	0.0503057
Total % success	99.9497

TOTAL percentage with OutGuess Hidden neurons: {15} functions: Sigmoid-Output_Hyperbolic_Tangent	
Cover total	5246
Cover errors	41
Cover % error	0.781548
Cover % success	99.2185
Stego total	135 891
Stego errors	49
Stego % error	0.0360583
Stego % success	99.9639
Total % error	0.0637678
Total % success	99.9362

TOTAL percentage with OutGuess Hidden neurons: {16} functions: Hyperbolic_Tangent-Output_Hyperbolic_Tangent	
Cover total	5246
Cover errors	27
Cover % error	0.514678
Cover % success	99.4853
Stego total	135 891
Stego errors	37
Stego % error	0.0272277
Stego % success	99.9728
Total % error	0.045346
Total % success	99.9547

TOTAL percentage with OutGuess Hidden neurons: {16} functions: Hyperbolic_Tangent-Output_Saturated_Linear	
Cover total	5246
Cover errors	7
Cover % error	0.133435
Cover % success	99.8666
Stego total	135 891
Stego errors	44
Stego % error	0.0323789
Stego % success	99.9676
Total % error	0.0361351
Total % success	99.9639

TOTAL percentage with OutGuess Hidden neurons: {16} functions: Sigmoid-Output_Sigmoid	
Cover total	5246
Cover errors	70
Cover % error	1.33435
Cover % success	98.6657
Stego total	135 891
Stego errors	64
Stego % error	0.0470966
Stego % success	99.9529
Total % error	0.0949432
Total % success	99.9051

TOTAL percentage with OutGuess Hidden neurons: {17} functions: Hyperbolic_Tangent-Output_Hyperbolic_Tangent	
Cover total	5246
Cover errors	22
Cover % error	0.419367
Cover % success	99.5806
Stego total	135 891
Stego errors	35
Stego % error	0.0257559
Stego % success	99.9742
Total % error	0.0403863
Total % success	99.9596

TOTAL percentage with OutGuess Hidden neurons: {17} functions: Hyperbolic_Tangent-Output_Sigmoid	
Cover total	5246
Cover errors	84
Cover % error	1.60122
Cover % success	98.3988
Stego total	135 891
Stego errors	16
Stego % error	0.0117741
Stego % success	99.9882
Total % error	0.0708531
Total % success	99.9291

TOTAL percentage with OutGuess Hidden neurons: {17} functions: Saturated_Linear-Output_Sigmoid	
Cover total	5246
Cover errors	5
Cover % error	0.0953107
Cover % success	99.9047
Stego total	135 891
Stego errors	85
Stego % error	0.0625501
Stego % success	99.9374
Total % error	0.0637678
Total % success	99.9362

TOTAL percentage with OutGuess Hidden neurons: {17} functions: Sigmoid-Output_Hyperbolic_Tangent	
Cover total	5246
Cover errors	19
Cover % error	0.362181
Cover % success	99.6378
Stego total	135 891
Stego errors	50
Stego % error	0.0367942
Stego % success	99.9632
Total % error	0.0488887
Total % success	99.9511

TOTAL percentage with OutGuess Hidden neurons: {17} functions: Sigmoid-Output_Saturated_Linear	
Cover total	5246
Cover errors	9
Cover % error	0.171559
Cover % success	99.8284
Stego total	135 891
Stego errors	22
Stego % error	0.0161894
Stego % success	99.9838
Total % error	0.0219645
Total % success	99.978

TOTAL percentage with OutGuess Hidden neurons: {18} functions: Saturated_Linear-Output_HyperbolicTangent	
Cover total	5246
Cover errors	29
Cover % error	0.552802
Cover % success	99.4472
Stego total	135 891
Stego errors	47
Stego % error	0.0345865
Stego % success	99.9654
Total % error	0.0538484
Total % success	99.9462

TOTAL percentage with OutGuess Hidden neurons: {18} functions: Sigmoid-Output_Hyperbolic_Tangent	
Cover total	5246
Cover errors	14
Cover % error	0.26687
Cover % success	99.7331
Stego total	135 891
Stego errors	66
Stego % error	0.0485683
Stego % success	99.9514
Total % error	0.0566825
Total % success	99.9433

TOTAL percentage with OutGuess Hidden neurons: {18} functions: Sigmoid-Output_Sigmoid	
Cover total	5246
Cover errors	19
Cover % error	0.362181
Cover % success	99.6378
Stego total	135 891
Stego errors	60
Stego % error	0.044153
Stego % success	99.9558
Total % error	0.055974
Total % success	99.944

TOTAL percentage with OutGuess Hidden neurons: {19} functions: Hyperbolic_Tangent-Output_Sigmoid	
Cover total	5246
Cover errors	54
Cover % error	1.02936
Cover % success	98.9706
Stego total	135 891
Stego errors	0
Stego % error	0.
Stego % success	100.
Total % error	0.0382607
Total % success	99.9617

TOTAL percentage with OutGuess Hidden neurons: {19} functions: Sigmoid-Output_Saturated_Linear	
Cover total	5246
Cover errors	4
Cover % error	0.0762486
Cover % success	99.9238
Stego total	135 891
Stego errors	0
Stego % error	0.
Stego % success	100.
Total % error	0.00283413
Total % success	99.9972

TOTAL percentage with OutGuess Hidden neurons: {19} functions: Sigmoid-Output_Sigmoid	
Cover total	5246
Cover errors	25
Cover % error	0.476554
Cover % success	99.5234
Stego total	135 891
Stego errors	66
Stego % error	0.0485683
Stego % success	99.9514
Total % error	0.0644764
Total % success	99.9355

TOTAL percentage with OutGuess Hidden neurons: {20} functions: Hyperbolic_Tangent-Output_Saturated_Linear	
Cover total	5246
Cover errors	17
Cover % error	0.324056
Cover % success	99.6759
Stego total	135 891
Stego errors	72
Stego % error	0.0529836
Stego % success	99.947
Total % error	0.0630593
Total % success	99.9369

Appendix B: Successful results for F5 algorithm

TOTAL percentage with algorithm F5 Hidden neurons: {1} functions: Hyperbolic_Tangent-Output_Sigmoid	
Cover total	9746
Cover errors	48
Cover % error	0.49251
Cover % success	99.5075
Stego total	77 314
Stego errors	26
Stego % error	0.0336291
Stego % success	99.9664
Total % error	0.0849989
Total % success	99.915

TOTAL percentage with algorithm F5 Hidden neurons: {2} functions: Saturated_Linear-Output_HyperbolicTangent	
Cover total	9746
Cover errors	13
Cover % error	0.133388
Cover % success	99.8666
Stego total	77 314
Stego errors	32
Stego % error	0.0413897
Stego % success	99.9586
Total % error	0.0516885
Total % success	99.9483

TOTAL percentage with algorithm F5 Hidden neurons: {3} functions: Sigmoid-Output_Saturated_Linear	
Cover total	9746
Cover errors	47
Cover % error	0.482249
Cover % success	99.5178
Stego total	77 314
Stego errors	24
Stego % error	0.0310422
Stego % success	99.969
Total % error	0.081553
Total % success	99.9184

TOTAL percentage with algorithm F5 Hidden neurons: {5} functions: Saturated_Linear-Output_HyperbolicTangent	
Cover total	9746
Cover errors	75
Cover % error	0.769546
Cover % success	99.2305
Stego total	77 314
Stego errors	0
Stego % error	0.
Stego % success	100.
Total % error	0.0861475
Total % success	99.9139

TOTAL percentage with algorithm F5 Hidden neurons: {10} functions: Saturated_Linear-Output_HyperbolicTangent	
Cover total	9746
Cover errors	8
Cover % error	0.082085
Cover % success	99.9179
Stego total	77 314
Stego errors	63
Stego % error	0.0814859
Stego % success	99.9185
Total % error	0.081553
Total % success	99.9184

TOTAL percentage with algorithm F5 Hidden neurons: {10} functions: Saturated_Linear-Output_Saturated_Linear	
Cover total	9746
Cover errors	7
Cover % error	0.0718243
Cover % success	99.9282
Stego total	77 314
Stego errors	64
Stego % error	0.0827793
Stego % success	99.9172
Total % error	0.081553
Total % success	99.9184

TOTAL percentage with algorithm F5 Hidden neurons: {12} functions: Saturated_Linear-Output_Saturated_Linear	
Cover total	9746
Cover errors	17
Cover % error	0.174431
Cover % success	99.8256
Stego total	77 314
Stego errors	24
Stego % error	0.0310422
Stego % success	99.969
Total % error	0.047094
Total % success	99.9529

TOTAL percentage with algorithm F5 Hidden neurons: {13} functions: Saturated_Linear-Output_HyperbolicTangent	
Cover total	9746
Cover errors	24
Cover % error	0.246255
Cover % success	99.7537
Stego total	77 314
Stego errors	48
Stego % error	0.0620845
Stego % success	99.9379
Total % error	0.0827016
Total % success	99.9173

TOTAL percentage with algorithm F5 Hidden neurons: {13} functions: Sigmoid-Output_Hyperbolic_Tangent	
Cover total	9746
Cover errors	36
Cover % error	0.369382
Cover % success	99.6306
Stego total	77 314
Stego errors	32
Stego % error	0.0413897
Stego % success	99.9586
Total % error	0.0781071
Total % success	99.9219

TOTAL percentage with algorithm F5 Hidden neurons: {15} functions: Saturated_Linear-Output_HyperbolicTangent	
Cover total	9746
Cover errors	8
Cover % error	0.082085
Cover % success	99.9179
Stego total	77 314
Stego errors	24
Stego % error	0.0310422
Stego % success	99.969
Total % error	0.0367563
Total % success	99.9632

TOTAL percentage with algorithm F5 Hidden neurons: {17} functions: Saturated_Linear-Output_Saturated_Linear	
Cover total	9746
Cover errors	22
Cover % error	0.225734
Cover % success	99.7743
Stego total	77 314
Stego errors	32
Stego % error	0.0413897
Stego % success	99.9586
Total % error	0.0620262
Total % success	99.938

TOTAL percentage with algorithm F5 Hidden neurons: {20} functions: Saturated_Linear-Output_Saturated_Linear	
Cover total	9746
Cover errors	28
Cover % error	0.287297
Cover % success	99.7127
Stego total	77 314
Stego errors	48
Stego % error	0.0620845
Stego % success	99.9379
Total % error	0.0872961
Total % success	99.9127

TOTAL percentage with algorithm F5 Hidden neurons: {20} functions: Sigmoid-Output_Saturated_Linear	
Cover total	9746
Cover errors	19
Cover % error	0.194952
Cover % success	99.805
Stego total	77 314
Stego errors	48
Stego % error	0.0620845
Stego % success	99.9379
Total % error	0.0769584
Total % success	99.923

Appendix C: Successful results for Steghide algorithm

TOTAL percentage with Steghide Hidden neurons: {1} functions: Saturated_Linear-Output_Saturated_Linear	
Cover total	5246
Cover errors	14
Cover % error	0.26687
Cover % success	99.7331
Stego total	142 772
Stego errors	4635
Stego % error	3.24643
Stego % success	96.7536
Total % error	3.14083
Total % success	96.8592

TOTAL percentage with Steghide Hidden neurons: {1} functions: Sigmoid-Output_Saturated_Linear	
Cover total	5246
Cover errors	32
Cover % error	0.609989
Cover % success	99.39
Stego total	142 772
Stego errors	3248
Stego % error	2.27496
Stego % success	97.725
Total % error	2.21595
Total % success	97.7841

TOTAL percentage with Steghide Hidden neurons: {2} functions: Saturated_Linear-Output_Saturated_Linear	
Cover total	5246
Cover errors	18
Cover % error	0.343119
Cover % success	99.6569
Stego total	142 772
Stego errors	3405
Stego % error	2.38492
Stego % success	97.6151
Total % error	2.31256
Total % success	97.6874

Appendix D: Successful results for PQ algorithm

TOTAL percentage with PQ Hidden neurons: {1} functions: Hyperbolic_Tangent-Output_Saturated_Linear	
Cover total	22 711
Cover errors	0
Cover % error	0.
Cover % success	100.
Stego total	126 599
Stego errors	2386
Stego % error	1.88469
Stego % success	98.1153
Total % error	1.59802
Total % success	98.402

TOTAL percentage with PQ Hidden neurons: {1} functions: Sigmoid-Output_Saturated_Linear	
Cover total	22 711
Cover errors	1
Cover % error	0.00440315
Cover % success	99.9956
Stego total	126 599
Stego errors	773
Stego % error	0.610589
Stego % success	99.3894
Total % error	0.518385
Total % success	99.4816

TOTAL percentage with PQ Hidden neurons: {3} functions: Hyperbolic_Tangent-Output_Sigmoid	
Cover total	22 711
Cover errors	4
Cover % error	0.0176126
Cover % success	99.9824
Stego total	126 599
Stego errors	1545
Stego % error	1.22039
Stego % success	98.7796
Total % error	1.03744
Total % success	98.9626

TOTAL percentage with PQ Hidden neurons: {3} functions: Sigmoid-Output_Saturated_Linear	
Cover total	22 711
Cover errors	0
Cover % error	0.
Cover % success	100.
Stego total	126 599
Stego errors	1612
Stego % error	1.27331
Stego % success	98.7267
Total % error	1.07963
Total % success	98.9204

TOTAL percentage with PQ Hidden neurons: {3} functions: Sigmoid-Output_Sigmoid	
Cover total	22 711
Cover errors	2
Cover % error	0.00880631
Cover % success	99.9912
Stego total	126 599
Stego errors	2615
Stego % error	2.06558
Stego % success	97.9344
Total % error	1.75273
Total % success	98.2473

TOTAL percentage with PQ Hidden neurons: {4} functions: Hyperbolic_Tangent-Output_Hyperbolic_Tangent	
Cover total	22 711
Cover errors	4
Cover % error	0.0176126
Cover % success	99.9824
Stego total	126 599
Stego errors	2469
Stego % error	1.95025
Stego % success	98.0497
Total % error	1.65629
Total % success	98.3437

TOTAL percentage with PQ Hidden neurons: {9} functions: Saturated_Linear-Output_HyperbolicTangent	
Cover total	22 711
Cover errors	0
Cover % error	0.
Cover % success	100.
Stego total	126 599
Stego errors	2555
Stego % error	2.01818
Stego % success	97.9818
Total % error	1.7112
Total % success	98.2888

TOTAL percentage with PQ Hidden neurons: {17} functions: Sigmoid-Output_Sigmoid	
Cover total	22 711
Cover errors	390
Cover % error	1.71723
Cover % success	98.2828
Stego total	126 599
Stego errors	1642
Stego % error	1.29701
Stego % success	98.703
Total % error	1.36093
Total % success	98.6391