Statistical Steganalysis of Images Using Open Source Software

Bhargavi Kaipa, Stefan A. Robila
Department of Computer Science
Montclair State University
Montclair, NJ 07043
robilas@mail.montclair.edu

***Abstract*— In this paper we present a novel steganalytic tool based on statistical pattern recognition. The main aim of our project was to design and implement a system able to classify the images into ones with no hidden message and steganographic images using classic pattern classification techniques such as Bayesian classification and decision trees. Experiments are conducted on a large data set of images to determine the classification algorithm that performs better by comparing classification success and error rates in each case. We have employed Weka, a data-mining tool developed in java for this purpose. We have also developed an application using Weka Java library for loading the data of the Images and classify the images into normal images and steganographic images. This application runs a GUI(Graphical User Interface) that enables the user to choose the classifier and other options required for the classification. Our results are aligned with current state of the art research and have the advantage of using open source software.**

Keywords- image processing, wavelets, steganography, steganalysis

I. INTRODUCTION *(HEADING 1)*

The word steganography is derived from the Greek words "steganos" and "graphein", which mean "covered" and "writing." Steganography is often confused with cryptography since both are used to protect important information. The difference between the two is that steganography involves hiding information in such a way that one cannot easily find that information is hidden at all [1]. Historical steganography involved techniques such as disappearing ink or microdots. Modern steganography involves hiding data in computer files [2]. It is fairly easy to hide a secret message in a graphic file without noticeably altering the visible appearance of that file.

In recent years, steganography has emerged as an increasingly active research area, with information being imperceptibly hidden in images, video, and audio among others. With the wide availability of digital images, and the high degree of redundancy present in them despite compression, there has been an increased interest in using digital images as cover-objects for the purpose of steganography. Steganalysis, i.e. the science of identifying steganographic data is thus increasing in importance.

While many approaches to steganalisys have been proposed recently, often such approaches come offered as commercial or freeware packages and do not allow any reasonable accuracy testing. In this paper, we describe the design and implementation of a system able to classify "non modified" and steganographic images using classic pattern recognition techniques such as Bayesian classification and decision trees.

II. STEGANOGRAPHY AND STEGANALYSIS

A generic description of the steganographic process can be given using the following formula :

$$cover\_medium + hidden\_data + stego\_key = stego\_medium \quad (1)$$

where the *cover_medium* is the file in which the data is hidden, which may also be encrypted using the *stego_key*.The resulting file is the *stego_medium*. The *cover_medium* (and, thus, the stego_medium) are typically image or audio files.

For example the cover image shown in Fig. 1. was embedded with the text of the Declaration of Independence (over 6,600 characters) to obtain the image on the right. For this we used the freely available Cryptobola Software that is able to embed encrypted data into jpeg images [3]. If presented with the stego-image, a human would not be able to realize that the image includes additional information. Even comparing side by side the stego and the cover medium would probably not help.

There are many steganographic techniques, including: concealing messages within the lowest bits of images or sound files, concealing data within encrypted data, and embedded pictures in video material etc. The simplest approach to hiding data within an image file is called *least significant bit* (LSB) insertion. In this method, we can take the binary representation of the hidden_data and overwrite the LSB of each byte within the cover image. If we are using 24-bit color, the amount of change will be minimal and indiscernible to the human eye [4].

Figure 1. Cover image on the left, stego-image on the right

978-1-4244-5550-8/10/$26.00 ©2010 IEEE

Though it is not possible to visually distinguish between the cover image and the steganographic image, they can be distinguished with the help of certain steganalysis techniques. Even if the secret content is not revealed, the existence of it can be detected since modifying the cover medium results in the change of its statistical properties and these changes cause distortions in the resulting steganographic medium's statistical properties. The distortions obtained can then be used to analyze and detect steganographic images [5].

In recent time, many steganographic algorithms have also been designed. Some of them are based on file type and others, which are more widely used are based on embedding method. Insertion-based techniques hide data in sections of a file that are ignored by the processing application and do not modify those bits that determine the contents of a file that are relevant to an end-user. In a substitution-based algorithm, the most insignificant bits of information that determine the meaningful content of the original file are replaced with new data in a way that causes the least amount of distortion [6].

Within steganographic techniques, a special concern is given to compressed information such as Jpeg pictures. The Jpeg file format is compact and does not significantly degrade the quality of an image so it is frequently used on the internet. The Jpeg format uses a discrete cosine transform (DCT) to identify 64 DCT coefficients in successive 8x8 pixel blocks. Of these quantized coefficients, the least significant bits are used to embed data. Because modifications to these bits affect pixel frequency as opposed to spatial structure no obvious distortion is present [7]. Most Jpeg Steganography methods replace the least significant bits (LSB) of the frequency coefficients (DCT) skipping 0s and 1s after quantization (e.g. J-Steg, JP Hide&Seek, OutGuess) [8]. Other methods decrement the coefficients' absolute values when the LSBs do not match, except coefficients with value zero (e.g. F3, F4, F5) [2].

Steganalysis is not an easy task because of the diversity of natural images and the wide variation of data embedding algorithms. However, an original cover medium and its stego-version (with hidden message inside) always differ from each other in some aspects since the cover medium is modified during the data embedding. Although all embedding techniques utilize redundancies in the cover image for the embedding process, they differ on their approach, and the image type they operate on. For example more recent techniques such as F5, Outguess, and Perturbed quantization embedding operate on Jpeg images by modifying the DCT coefficients. Although changing the DCT coefficients will cause unnoticeable visual artifacts, they do cause detectable statistical changes. These statistical changes are used by steganalysis techniques to detect any embedded messages.

Essentially there are two approaches to the problem of steganalysis. One is to come up with a steganalysis method specific to a particular steganographic algorithm. The other is developing universal steganalysis techniques which are independent of the steganographic algorithm being analyzed [7]. Each of the two approaches have their own advantages and disadvantages. A steganalysis technique specific to an embedding method would give very good results when tested only on that embedding method, and might fail on all other steganographic algorithms. On the other hand, a steganalysis method which is independent of the embedding algorithm might perform less accurately overall but still provide acceptable results on new and unseen embedding algorithms. Based on whether an image contains hidden message, images can be classified into two classes: the image with no hidden message and the corresponding steganogrphic image (the very image but with message hidden in it). Steganalysis can thus be considered as a pattern recognition process to decide which class a test image belongs to. The key issue for steganalysis just like for pattern recognition is feature extraction. The features should be sensitive to the data hiding process. In other words, the features should be rather different for the image without hidden message and for the steganographic image. The larger the difference, the better the features are. The features should be as general as possible, i.e., they are effective to all different types of images and different data hiding schemes [2].

III. PROPOSED METHOD

A. *Overall Approach*

The Methodology that is adopted for the project implementation has the sequence flow as shown in Fig. 2. A collection of stego and plain images is being first processed for extraction of features. The features are then reduced in number through an extraction process and the reminaingin values are fed to a classifier.

B. *Feature Selection*

Jpeg steganographic algorithms usually embed message bits into randomly chosen DCT coefficients. Hence the histogram of the DCT coefficients of a stego image is different from the histogram of the DCT coefficients of the cover image, allowing the selection of the coefficients as features. Fig. 3 provides an example of a cover and stego image with their corresponding DCT coefficient histograms.

The features were computed using the tool ImageJ. ImageJ is an open source image processing tool written in Java. It is downloadable on any computer with Java 1.4 or later virtual machine [9]. In the current format, our application requires manual processing of each image through ImageJ. However, it can be easily envisioned that the DCT and the histogram coefficients can be computed automatically without human intervention.

Figure 2. Classification of steganographic images

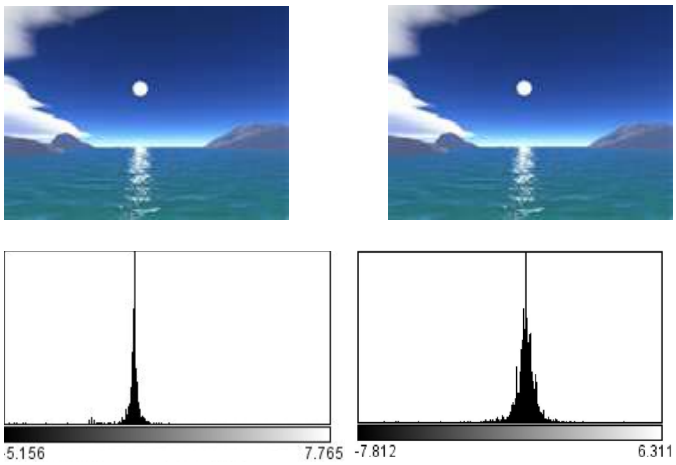

Figure 3. Cover image on the left, stego-image on the right and the corresponding DCT histograms below.

C. Classification

Classification of the images is done using Weka [10]. Weka stands for Waikato Environment for Knowledge Analysis. Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes. Weka is developed by the machine-learning group of Computer Science Department, University of Waikato, New Zealand .

To classify we used the standard supervised classification approach. Here, the set of images is split in training and validation subsets. The training subset together with information on whether they are cover or stego images is used to train the classifier. The task of the supervised classifier is to predict the label (stego or cover) for any image after having seen a number of training examples (i.e. pairs of input and target output). To achieve this, the classifier has to generalize from the presented data to new possible inputs [11]. Two different classification approaches were employed. The first uses Naïve Bayesian classification while the second uses decision trees.

The *naïve Bayes classifier* is based on the Bayesian theorem. It is particularly suited when dimensionality of the inputs is high. Parameter estimation for naive Bayes models uses the method of maximum likelihood. In spite of over simplified assumptions, it often performs better in many complex real-world situations. The main advantage is that it requires a small amount of training data to estimate the parameters. In a nutshell, the classifier works as follows:

Given a set D of n dimensional vectors $\mathbf{x}$ ($x_1, x_2, x_3, \ldots, x_n$), and m classes : $C_1, C_2, \ldots, C_m$ the Naïve Bayes classifier predicts $\mathbf{x}$ belongs to class C_m if:

$$P(C_i|\mathbf{x}) > P(C_j|\mathbf{x}) \text{ for all j between 1 and } m \qquad (2)$$

The above conditional probability can be expressed using the Bayes theorem:

$$P(C_i \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid C_i)P(C_i)}{P(\mathbf{x})} \qquad (3)$$

As P(x) is constant, eq. 2 reduces to maximizing

$$P(\mathbf{x} \mid C_i)P(C_i) \qquad (4)$$

While this approach is computationally expensive for large n, to ease the burden "class conditional independence" is assumed resulting in:

$$P(\mathbf{x} \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i) \qquad (5)$$

Overall, naïve Bayes classification performs extremely well for a large number of problems. It's simplicity ensures widespread use and the relatively small number of exemplars needed for training is another supporting factors. Nevertheless, recent studies have shown that Bayesian classification is outperformed by most modern classifiers [12].

The second approach is based on decision trees. A decision tree is a predictive model that maps observations about an item with conclusions about its target value. The machine learning technique for inducing a decision tree from data is called decision tree learning [13]. More descriptive names for such tree models are classification tree (discrete outcome) or regression tree (continuous outcome). In these tree structures, leaves represent classifications and branches represent conjunctions of features that lead to those classifications. The machine learning technique for inducing a decision tree from data is called decision tree learning, or decision trees.

Decision trees are hierarchical tree structures that can be used to classify based on a series of questions (or rules) about the attributes of the class. The attributes of the classes can be any type of variables from binary, nominal, ordinal, and quantitative values, while the classes must be qualitative type (categorical or binary, or ordinal). In short, given a data of attributes together with its classes, a decision tree produces a sequence of rules (or series of questions) that can be used to recognize the class. There are several most popular decision tree algorithms such as ID3, C4.5 and CART (classification and regression trees). In general, the actual decision tree algorithms are recursive. In our case we used C4.5 developed by Ross Quinlan [14], as an extension of an earlier approach (ID3). The decision trees generated by C4.5 can be used for classification, and for this reason, C4.5 is often referred to as a statistical classifier. C4.5 builds decision trees from a set of training data, using the concept of information entropy.

IV. APPLICATION

Weka Java library is leveraged to develop the application. All the classes needed for the application are imported from Weka Java library to our application. Java Code is written in Eclipse IDE to load the data into the application, build the classifier and evaluate the classifier. GUI is also developed in

eclipse IDE. Three sets of classes are needed for the application: loading data, building the classifier and evaluating it. Our GUI based interface allows us to load the features for a set of images and train the classifier using either naïve Bayes or decision trees. Next, the application allows classification of new images using either of the classifiers.

V. RESULTS

A data set of 1400 images is created by first taking 900 images from the web. To create the setgo images, we devlopped a java application that generated random length text and embedded it using the F5 algorithm [15], resulting in 500 stego images.

Experiments are conducted by repeating 10 times the following: split the data in 70%-30% ratio for train/ test data that is 70% of the data is used for training the classifier and the remaining 30% is used for testing. The results obtained are in Tables 1 and 2. The average results are provided in Fig. 4. Decision tree classifier performed consistently well by correctly classifying up to 70%-73% of the images. From the results obtained from Weka as well as our application we can say that the performance of decision tree classifier is better than naïve Bayes classifier.

To further understand the ability of each approach to classify correctly we looked at the confusion matrix. A confusion matrix contains information about actual and predicted classifications done by the classification system [16]. Performance of the classifier can be evaluated using the data in the matrix. The entries in the confusion matrix have the following meaning in the context of our study: *a* is the fraction of correct predictions that an instance is negative, *b* is the fraction of incorrect predictions that an instance is positive, *c* is the fraction of incorrect predictions that an instance negative, and *d* is the fraction of correct predictions that an instance is positive (see Table 3). Table 4 presents the confusion matrices for both approaches. The data suggest that naïve Bayes is biased towards detection of stego images while decision trees is more accurate for cover images.

VI. CONCLUSION

A system that can classify the images into images with no hidden message and steganographic images using classic pattern recognition techniques such as Bayesian classification and decision trees was designed, developed and implemented successfully.

Experiments were conducted on a large data set of images to determine the classification algorithm that performs better by comparing classification success and error rates in each case. We have employed Weka, a data mining tool developed in java for this purpose. We have also developed an application using Weka Java library for loading the data of the Images and classify the images into normal images and steganographic images. This application runs a GUI (Graphical User Interface) that enables the user to choose the classifier and other options required for the classification.

TABLE I. NAÏVE BAYES RESULTS

	Correctly classified Instances		Incorrectly classified instances	
1.	150	35.8852 %	268	64.1148 %
2.	138	33.0144 %	280	66.9856 %
3.	148	35.4067 %	270	64.5933 %
4.	129	30.8612 %	289	69.1388 %
5.	154	36.8421 %	264	63.1579 %
6.	142	33.9713 %	276	66.0287 %
7.	135	32.2967 %	283	67.7033 %
8.	154	36.8421 %	264	63.1579 %
9.	130	31.1004	288	68.8995%
10.	131	31.3397 %	287	31.3397 %

TABLE II. DECISION TREE CLASSIFIER RESULTS

	Correctly classified Instances		Incorrectly classified instances	
1.	286	68.4211 %	132	31.5789 %
2.	306	73.2057 %	112	26.7943 %
3.	277	66.2679 %	141	33.7321 %
4.	284	67.9426 %	134	32.0574 %
5.	272	65.0718 %	146	34.9282 %
6.	289	69.1388 %	129	30.8612 %
7.	298	71.2919 %	120	28.7081 %
8.	319	76.3158 %	99	23.6842 %
9.	282	67.4641 %	136	32.5359 %
10.	295	70.5741%	123	29.4258%

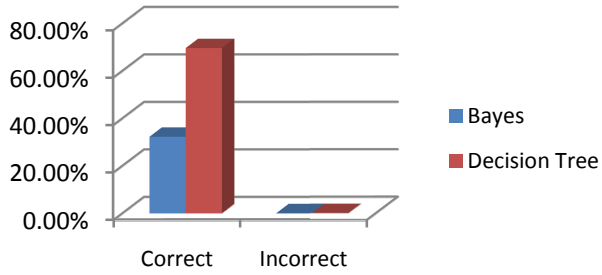

Figure 4. Average classification results.

TABLE III. CONFUSION MATRIX

		Predicted	
		Negative	Positive
Actual	Negative	**a**	**b**
	Positive	**c**	**d**

TABLE IV. CONFUSION MATRIX

	Naïve Bayes		Decision Tree	
	Predict Cover	Predict Stego	Predict Cover	Predict Stego
Real Cover	**13.34%**	86.66%	**87.27%**	12.73%
Real Stego	20.73%	**79.27%**	63.41%	**36.59%**

Steganography algorithms continue to evolve and add new deviations such as increased encryption strength, varied message length or random positioning of the data. At each step, steganalysis techniques focused on a single approach become obsolete since they cannot detect the new

steganographic methods. Techniques like the one we provide are considered "blind", i.e. independent of the steganographical approach, and hold the very exciting potential to transcend the current fragile nature of modern steganalysis. They allow the detection that a file has a hidden message, even if it is hidden using a new, previously unseen steganography algorithm since some or the other alterations are made by these algorithms after embedding the data. While not highly accurate, blind steganalysis remains an irreplaceable tool in the chase with recent steganography approaches.

REFERENCES

[1] Y. Wang and P. Moulin, "Optimized feature extraction for learning-based image steganalysis," *IEEE Transactions on Information Forensics and Security*, vol. 2, 2007, pp. 31–45.

[2] X.Y. Luo, D.S. Wang, P. Wang, and F.L. Liu, "A review on blind detection for image steganography," *Signal Processing*, 2008.

[3] CryptoBola, *CryptoBola*, http://www.cryptobola.com/index.htm: .

[4] J. Zhang, Y. Hu, and Z. Yuan, "Detection of LSB Matching Steganography using the Envelope of Histogram," *Journal of Computers*, vol. 4, 2009, p. 647.

[5] C. Zhou, J. Feng, and Y. Yang, "Blind Steganalysis Based on Features in Fractional Fourier Transform Domain."

[6] M. Weiss, "Principles of Steganography."

[7] M. Kharrazi, H.T. Sencar, and N. Memon, "Image steganography: Concepts and practice," *Lecture Note Series, Institute for Mathematical Sciences, National University of Singapore*, 2004.

[8] G. Berg, I. Davidson, M.Y. Duan, and G. Paul, "Searching for hidden messages: Automatic detection of steganography," *Proceedings of the 15th Innovative Applications of AI Conference*, 2003, pp. 51–56.

[9] M.D. Abramoff, P.J. Magalhaes, and S.J. Ram, "Image processing with ImageJ," *Biophotonics International*, vol. 11, 2004, pp. 36-43.

[10] I.H. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes, and S.J. Cunningham, "Weka: Practical machine learning tools and techniques with Java implementations," *ICONIP/ANZIIS/ANNES*, Citeseer, 1999, pp. 192-196.

[11] S.B. Kotsiantis, "Supervised machine learning: A review of classification techniques," *EDITORIAL BOARDS, PUBLISHING COUNCIL*, vol. 31, 2007, pp. 249-268.

[12] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," *Proceedings of the 23rd international conference on Machine learning*, ACM, 2006, p. 168.

[13] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern classification*, Citeseer, 2001.

[14] J.R. Quinlan, *C4. 5: programs for machine learning*, Morgan Kaufmann, 2003.

[15] J. Fridrich, M. Goljan, and D. Hogea, "Steganalysis of JPEG images: Breaking the F5 algorithm," *Lecture notes in computer science*, 2003, pp. 310-323.

[16] F. Provost, T. Fawcett, and R. Kohavi, "The case against accuracy estimation for comparing induction algorithms," *Proceedings of the Fifteenth International Conference on Machine Learning*, Citeseer, 1998, pp. 445-453.