STEGANALYSIS OF DIGITAL IMAGES USING RICH IMAGE REPRESENTATIONS AND ENSEMBLE CLASSIFIERS

BY

JAN KODOVSKÝ

MS, Czech Technical University, Prague, 2006

DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate School of Binghamton University
State University of New York
2012

Accepted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate School of Binghamton University
State University of New York
2012

March 8, 2012

Jessica Fridrich, Chair and Faculty Advisor
Department of Electrical and Computer Engineering, Binghamton University

Mark Fowler, Member
Department of Electrical and Computer Engineering, Binghamton University

Zhongfei (Mark) Zhang, Member
Department of Computer Science, Binghamton University

Siwei Lyu, Outside Examiner
Department of Computer Science, University at Albany

Abstract

Modern detectors of steganographic communication in digital images are implemented as supervised classifiers trained in pre-defined feature spaces also called image models. Currently, the Support Vector Machine (SVM) is the machine-learning tool of choice in the steganalysis community due to its accuracy and a well-founded theory. However, in order to keep the SVM training computationally feasible, feature spaces need to be designed to be low-dimensional. Consequently, their construction often consists of a series of clever tricks and heuristic dimensionality-reduction techniques. Recent trends in steganalysis, however, have shown that more complex and higher-dimensional image models could deliver substantially better performance.

In this dissertation, we propose a novel framework for steganalysis of digital images in which we replace SVMs with the ensemble classifier, a scalable machine-learning alternative offering comparable accuracy at a fraction of a computational cost. This allows us to approach the feature-space building process in a more systematic and exhaustive way. In particular, we propose to construct feature spaces as collections of a large number of simpler submodels, each of them capturing different types of dependencies among image coefficients. As a result, we obtain a high-dimensional rich statistical descriptor of images, the so-called *rich model*, which is then used for the classifier training.

To demonstrate the power of the proposed methodology, we construct rich models for images in raster formats and JPEGs, the two most commonly used image representations. The $34,671$-dimensional spatial domain rich model consists of co-occurrences of neighboring samples of noise residuals obtained by various filters, and the $22,510$-dimensional JPEG domain rich model consists of submodels capturing different types of dependencies among DCT coefficients of JPEG images. Both rich models, combined with the ensemble classifier, are shown to significantly outperform previous art across a wide range of steganographic schemes hiding data in both domains.

This work is presented as a self-contained text, covering all technical details of the ensemble classifier and the rich-model construction, their implementation, and experimental performance evaluation.

Acknowledgements

I would like to express my deepest gratitude to my PhD adviser, Jessica Fridrich, for her excellent guidance throughout my doctoral years. She provided me with a very pleasant research atmosphere, exposed me to the joy of academic career, and helped me find a proper balance between my personal and professional goals. I am indebted to her for years of encouragement and inspiration, for sharing the excitement of scientific discoveries, for memorable Thanksgiving evenings, for friendship.

I am very grateful to the members of my dissertation committee and to the professors of Binghamton University who provided me with professional knowledge and background needed for this thesis. I thank Scott Craver for his contagious enthusiasm for digital security, and Mark Fowler for his challenging homeworks and an intriguing problem for my comprehensive exam. I would also like to thank the students of my EECE 580B class for their valuable feedback and for what they taught me.

My gratitude is extended to the current and past members of the Digital Data Embedding laboratory, to the temporary visiting scholars, and to the colleagues from other departments at Binghamton University who helped me on the way towards my PhD degree. I want to give a special thanks to Paul Blythe and Jan Lukáš for their friendship and help in the first years of my studies, and to Matthias Kirchner for inspirational discussions over the countless cups of coffee during his visit.

It is my pleasure to also acknowledge my graduate friends and the members of European Student Association who all made my time in Binghamton so memorable. I particularly thank Abhinesh Bhuvanesh for his friendship and a true Indian cuisine experience, and Jan Konečný for the times of glory during 'unimacing'. I am also thankful to my friends from Oceanography Department at Texas A&M University with whom I spent my final doctoral year.

I am greatly indebted to my very dear family, to my parents, grandparents, and to my sister Martina. My doctoral studies took me far away from home and I am grateful for their understanding, for their great support, and for being there when I needed them.

My most heartfelt acknowledgment goes to my girlfriend Fenix who stayed next to me during my PhD pursuit. I express my gratitude, love, and sincere appreciation for her emotional support, for trusting me, and for loving me.

This work would not have been possible without the continuous financial support from the Air Force Office of Scientific Research. In particular, my work was funded from the research grants FA8750-04-1-0112, FA9550-08-1-0084, and FA9550-09-1-0147.

Jan Kodovský

Binghamton, 2012

Contents

List of Tables

List of Figures

List of Algorithms

Preface

Private communication has been an active research area for many decades and experienced a boom at the turn of the 21st century as digital representation of information took over its analog predecessor and the Internet use became widespread. The privacy of digital communication over an insecure or a monitored communication channel is often associated with cryptography, which transforms a given message into an unintelligible text. This ciphertext can then be transmitted over a public channel as the message cannot be decrypted (at least not in a reasonable time) without a valid crypto-key possessed only by trusted parties.

Steganography adds an additional layer of security as it hides the very fact that a secret communication takes place. It conceals the encrypted message into an ordinary looking traffic and only the intended recipient is aware of its existence. Using a proper stego-key, the hidden message can be extracted and eventually decrypted. Even though steganography is an art with roots in ancient times, it has not been until the last two decades that it has become a rigorous research discipline attracting many scientists from different fields, such as information theory, computer science, and signal processing. Nowadays, there exist hundreds of steganographic tools capable of hiding information into electronic documents and digital media files, such as audio, video, and still images.

Naturally, parallel to the efforts of hiding information into innocuously looking digital objects, scientists have been trying to develop techniques for detecting this secret information exchange; that is the goal of steganalysis. While a cryptographic protocol is considered broken when there exists an attack that is computationally faster than a brute force search, a steganographic scheme is considered broken when there exists a mechanism that can distinguish between the clean (cover) channel traffic and the one containing secret communication with probability better than random guessing. In other words, a mere raised suspicion of a secret information exchange between the sender and the recipient indicates the failure of steganography regardless of the fact whether or not an attacker is able to extract the actual content of the secret message.

This PhD thesis is about steganalysis of digital images, currently the most common carriers of secret information for steganography. There are several reasons why multimedia files serve as ideal covers for steganography. First, they are commonly being exchanged among millions of Internet users on a daily basis, creating a natural high traffic information channel – think of any image sharing portal or a social networking website. Secondly, media files are usually high volume data and therefore offer a large space for data hiding. Finally, media files contain a lot of noise and other indeterministic components that could be easily adapted for information hiding without degradation of the quality of the original media and without leaving perceivable traces. Digital images are particularly popular steganographic carriers among other multimedia files as digital photography and cell-phone cameras have become part of our everyday life while sharing images over the Internet is now a matter of a few clicks. In fact, most of the existing steganographic tools hide information into digital images and image steganography is the most advanced steganographic branch.

Fundamentally, steganalysis is a hypothesis-testing problem as a steganalyst needs to determine whether or not the given image contains a hidden message. However, even though we could theoretically construct optimal steganalysis detectors as likelihood-ratio tests, this is in practice impossible as the dimensionality and the complexity of cover objects (digital images) is too high. Any attempt

to introduce a simplified low-dimensional model must be inevitably inaccurate. Therefore, the task of steganalysis is usually viewed as a classification problem and approached using machine learning tools; a popular classifier in the steganalysis community is the support vector machine (SVM).

A typical steganalysis detector works in two stages. In the first stage, a set of statistical quantities (features) is extracted from a prepared database of cover images and stego images (images containing a secret message). An example of features may be the histogram of pixel values from the entire image. Once the features are extracted, the steganalyst proceeds to the second stage, a supervised binary classification, where a classifier is trained on a portion of the features and its accuracy is evaluated on the rest. The feature space, i.e., the space formed by the extracted features, is the central concept in steganalysis as it is where the classification takes place. It is a low-dimensional representation of images and it needs to be carefully designed, specifically for the purpose of steganalysis. Understanding feature spaces, their construction and intrinsic properties is of utmost importance for a successful steganalyzer design.

The inability to accurately model digital images determines how modern steganographic methods are constructed. Rather than trying to mimic or preserve some generic image model, message is hidden by minimizing the impact of embedding measured by an appropriately defined distortion function, which can be, in the simplest case, the number of embedding changes. Boosted by sophisticated coding schemes, this paradigm gave birth to the most secure steganographic methods up to date. Steganalysts need to take this trend into account in the feature space design – features need to be sensitive to the very subtle steganographic modifications. At the same time, in order to reduce the large variance of features across different images, the influence of the image content should be suppressed as much as possible. A good feature space usually captures different dependencies among image coefficients (pixels in case of images in raster format or DCT coefficients for JPEG images) as these are often disturbed during message embedding.

Many different feature sets have already been proposed for steganalysis of different steganographic schemes. And it seems that it is always only a matter of time when a security of a newly proposed steganographic scheme is compromised by a newly designed feature set. In a sense, the never-ending battle between steganography and steganalysis is a race for a better (or more complete) model of statistical dependencies among individual cover elements. A steganographer tries to reflect this model into his or her distortion function while a steganalyst aims to improve upon the steganographer's model by finding those dependencies that were omitted and construct the features from them.

The competition between steganography and steganalysis creates a self-stimulating environment that is beneficial to the field, while, to an inexperienced observer, it may seem that there is never going to be a clear winner. There is one important factor, however, favoring steganography – the growing dimensionality and complexity of the model. While the distortion function can be easily defined and calculated in spaces of a very high dimension (in the order of millions or more), the dimensionality of feature spaces needs to be kept low, typically in the order of hundreds and rarely exceeding 1,000. The reason is the involved machine learning and the threat of the curse of dimensionality – a growing computational complexity of training and an insufficient number of training samples. Consequently, the design of feature spaces is constrained to low dimensions. This did not seem to be a limitation for the early steganalysis algorithms whose feature space dimensions hardly exceeded 100, but, as the level of sophistication of steganographic algorithms grew, the high dimensionality became a major obstacle. Consequently, the design of feature spaces for steganalysis has become an increasingly more challenging discipline as not only do steganalysts need to cover a wide spectrum of dependencies but they also need to combine them together using clever tricks and advanced dimensionality reduction techniques, to end up with an image representation that would be low-dimensional and perform well against modern steganographic algorithms.

In this thesis, we challenge the standard approach of today's steganalysis and propose a novel and clean framework for steganalysis of digital images. The goal is to introduce a methodology that would not be constrained to low-dimensional image representations, a limitation we see as a major obstacle for future development of steganalysis. Furthermore, we aim to automatize feature-based steganalysis as much as possible – to formulate a framework that would create an optimal feature

space for a given (new) steganographic method, and thus remove the major bottleneck of steganalysis, which is the handcrafted feature space development.

The proposed framework can be divided into three parts:

1. We construct a "rich image representation," a large collection of various statistical descriptors of different dependencies among individual cover elements. This can be realized for example by merging many smaller submodels, each of them capturing a different type of statistical dependencies. The goal is to make the joint model (image representation) as complete as possible, with no dimensionality constraints.

2. The second step is classification. In order to use the constructed model for classification, we need to replace standard machine learning tools (SVM) with new ones that better scale w.r.t. the number of training samples and model dimensionality. We use ensemble classifiers for this purpose.

3. Finally, we introduce a methodology of a joint model assembly and classification. For a given training data (cover images and stego images of a specific steganographic method), we take the original rich image representation and identify only those sub-components that are effective for detection of the given scheme. This is a step towards automatization of (targeted) steganalysis and can be viewed as a dimensionality reduction or a feature selection technique with a classification feedback.

Even though parts of this dissertation were published as full-length papers at various conferences and journals, this is a self-contained text targeted at a generally knowledgeable person from the field. The thesis is organized as follows.

In the first chapter, we informally introduce steganography and steganalysis, put it into historical context, and motivate our work by mentioning a few recent cases of modern steganography used for malicious purposes. We talk about the role of steganalysis and its different types. The notation used in the thesis is introduced at the end of the chapter.

In Chapter 2, we formalize the concept of the steganographic channel and introduce an information-theoretic definition of steganographic security. The focus of the chapter is then shifted to steganalysis, its connection to signal detection, and practical realization through supervised classification. A historical perspective on the development of feature-based steganalysis is also provided.

A key component of our framework is a scalable machine learning tool. We propose to use ensemble classifiers that are built from a set of simpler detectors which could be trained easily. The learning algorithm is introduced in Chapter 3, where we also illustrate how its performance depends on the hyper-parameters. An algorithm for an automatic optimization of these parameters is then described, making the training fully automatized. The complexity of the ensemble classifier is experimentally compared with the one of SVM.

The purpose of Chapter 4 is to understand the fundamental principles of feature space design. We provide general guidelines for feature construction and demonstrate them on existing feature sets. We discuss the important technique of calibration and its evolution from its early implementation up to the so-called Cartesian calibration, which can be viewed as a method of model enrichment. As it is important to be aware of potential pitfalls, the chapter is concluded by a series of specific examples of imperfect feature spaces with various security weaknesses.

Equipped with a scalable machine learning and the insight about the inner workings of feature spaces, the PhD thesis escalates in Chapters 5 and 6 where we construct complex feature spaces (called rich models) for steganalysis of two image representations: raster formats (for example PNG, PGM, BMP, or TIFF) and the JPEG format, the most commonly used image format up to date. Rich image representations and ensemble classification are also combined into a single machinery for determining a subset of the model responsible for most of the detection accuracy for a given steganographic technique. Identification of the effective submodels provides a valuable insight into

the steganographic scheme, identifies its weaknesses, and consequently gives rise to an improved hiding technique, further advancing the field.

The thesis is concluded in Chapter 7, where we summarize the contributions and outline future directions for steganalysis.

Throughout the thesis, the concepts and ideas are demonstrated on various steganalysis experiments targeted at existing steganographic methods. The description of all steganalyzed steganographic schemes and used image datasets appears in Appendices A and B, respectively. Finally, Appendix C discusses important practical issues that should be considered when performing feature-based steganalysis experiments, for example the negative effects of a JPEG compressor and the header on the detection accuracy for certain steganographic schemes, or the importance of steganalysis-specific properties of feature spaces and the implications for cross-validation.

Chapter 1

Introduction

Any communication system that aims to conceal the very fact that the communication takes place can be classified as steganography. As the desire to communicate secretly accompanies our civilization from its ancient times, examples of steganography can be found throughout the history. However, these were more clever tricks and intriguing ideas rather than rigorously designed systems, and their success was usually based on the assumption that the eavesdropper was unaware even of the possibility of a secret information exchange – security through obscurity. Therefore, we cannot talk about steganalysis. We will discuss some of these earlier steganographic techniques in Section 1.1.

In Section 1.2, we present the popular prisoners' problem formulated by Simmons [124], one of the first attempts to formalize steganography. We introduce the three fictitious characters of Alice, Bob, and Eve, and discuss their goals and roles in the steganographic system. The role of Eve, the eavesdropper, will be of a particular interest to us as she represents the steganalyst, further discussed in Section 2.3.

With the ever-increasing technological advancements and their impacts on our society, especially the growing power of the Internet and the trends of on-line sharing and social networking, the potential threat of steganography being used for malicious purposes rapidly increases. Today's carriers of secret information are digital images and audio files, phone conversations and Voice over IP protocols, and even the web traffic itself. Modern digital steganography has been used by extremists and terrorists for spreading propaganda and co-ordinating their activities, by spies for secret communication with their headquarters, and by employees of engineering companies for stealing sensitive information. We will mention a few specific cases of these malicious activities in Section 1.4, stressing the importance of developing steganalysis techniques and further motivating our work.

The notation used throughout the dissertation is introduced in Section 1.5.

1.1 Historical perspective

The word steganography is of the Greek origin and translates as "covered writing." It works by hiding messages into innocuous carriers called *cover* objects. A cover object may be basically anything, for example a physical text document, a painting, or a piece of wood, as long as it can be used to deliver a hidden message to the intended recipient without raising suspicion of untrusted parties. Interestingly, the first documented cover object used for the purposes of steganography was the human body. According to Herodotus [56], Histiæus, the ruler of the ancient Greek city of Miletus in the 6th century BC, tattooed the message on a shaved head of his slave, covering it by the regrown hair. Another example of steganography, also mentioned in [56], was a wax-covered message scraped on the surface of a wooden writing table.

Apart from these ancient and rather amusing examples of a steganographic communication, the most popular covers were text documents. One possibility is to use the paper itself as a carrier and write the secret message between the lines of the official text using an invisible ink, such as milk, vinegar, or other organic fluids, making the message disappear after the paper dries and re-appear once heated above a candle. The other option is to hide the secret message *within* the text itself – for instance by inconspicuously marking the letters forming the secret message. There are numerous ways of doing so, for example by modifying the height of the appropriate letters [128], by using different fonts [4], or slightly shifting letter positions [12].

A popular way of conveying an additional message within a text, even though not truly steganographic as its presence was often known, was the so-called *acrostic*. Acrostic is a text (usually a poem) whose first letters or syllables of each line form a secret message. A famous example is Boccaccio's poem *Amorosa visione*, which contains an additional sonnet encoded as the first letters of every verse triplet [134]. A steganographic version of acrostic is a grille cipher and its future variants, originally invented by the Italian mathematician Girolamo Cardano in the 16th century. Here, the message is read by putting a mask over the text document, covering most of the letters and leaving only those forming the secret message. This was a significant security improvement over the previously mentioned letter-marking techniques, as the secret message could not be read without the mask shared between the two communicating parties. The likelihood of a raised suspicion depended solely on the linguistic skills of the sender as he needed to compose a real-looking and trustful cover text with given letters at certain positions.

An interesting steganographic technique is to shrink the text into very small dimensions so that it literally cannot be seen without a magnifying device. This idea was originally proposed by Brewster [15] and later technologically advanced into a secret communication system that was successfully used by spies during war conflicts of the 20th century. The resulting shrinked object, called "microdot," can be communicated for example under the stamp of an ordinary looking postcard or as a period at the end of a sentence.

Almost all steganographic techniques mentioned so far (with the exception of Cardano's mask-dependent cipher) are based on the assumption that the eavesdropper not only lacks the knowledge of the system the two communicating parties use, but is often also unaware of the mere *possibility* of a secret communication. This is called *security through obscurity*, and the experience has shown that it is always only a matter of time until the information about the system leaks to the attacker (or is reverse-engineered). As an example, let us mention the A5/1 cipher used in contemporary GSM networks to encrypt phone conversations. This cipher was originally kept secret, but eventually became publicly known and a number of serious security weaknesses have been found [8].

Since the only thing necessary to discover the presence of early steganographic systems was simply to possess proper knowledge, nothing like targeted breaking (or steganalysis) existed. Instead, trained spies and various espionage techniques were used to obtain the needed information. Arguably, the beginnings of steganalysis as a research discipline could be accounted to Simmons and his famous prisoners' problem formulated in 1983 [124].

1.2 The prisoners' problem

Two criminals, Alice and Bob, have been arrested and locked in separate cells. The warden Eve allows them to exchange messages but the communication has to be completely open to her. Since any suspicion of a secret information exchange would result in an immediate communication cut off, the prisoners cannot protect their messages by encrypting them. As Alice and Bob need to coordinate their escape plan, they need to find a different way to communicate secretly without being caught.

The task of Alice and Bob is steganography – their goal is to secretly communicate through a monitored channel without being detected. The job of the warden Eve is steganalysis – she needs to

find out whether or not Alice and Bob secretly exchange messages. Note that, unlike in cryptanalysis, Eve is not required to find out the actual content of the exchanged communication. The mere suspicion of the hidden communication would trigger an immediate cut of the communication channel – the failure of steganography.

Additionally, it is assumed that Eve has a complete knowledge of the steganographic algorithm that the prisoners may use. This is a very important assumption known as the Kerckhoffs' principle and it common in cryptography. It states that the security of the system should not be compromised even when everything about the system, except the key, is publicly known. This allows Eve to study impacts of the steganographic embedding on cover objects and to develop rigorous detection techniques. It also puts steganalysis on firm grounds.

1.3 Warden's options

Typically, Eve is assumed to be a *passive warden*, which means that she is not allowed to intervene in the communication between Alice and Bob. As this dissertation focuses on steganalysis of digital images, we will assume that Alice and Bob are exchanging digital photographs.[1] A passive warden is allowed to inspect the image Alice sends to Bob and even apply some statistical tests to the pixel values, etc. However, she is not allowed to modify the image in any way and has to pass it to Bob unaltered.

The passive warden scenario is the most studied steganalysis setting as Eve's goal is to detect the mere *presence* of the secret communication. Furthermore, it is reasonable to assume that Alice has a full control over the communication channel – let us reformulate the prisoners' problem with a passive warden in a modern (and more realistic) setting. Rather than prisoners confined in separate cells, Alice and Bob are citizens of different countries and want to secretly exchange sensitive information regarding their government's practices. Alice embeds her message into a photograph of her hamster and posts the picture to her on-line photo gallery that she regularly updates. Bob, as well as hundreds of other fans of hamsters, visits Alice's gallery and checks her newest photos. However, only Bob knows the true reason behind the photographs and uses the secret key to extract the hidden information. Eve does not have the control over the computer server Alice is using and therefore cannot modify the pictures in any way. However, she can download them and run her steganalysis tests.

An *active warden* is allowed to slightly modify the communicated objects in order to prevent Alice and Bob from steganographic communication. In case of digital images, she may for example compress the image using the JPEG format and/or apply some common image processing, such as gamma correction. This significantly complicates Alice's job as now her technique must be robust against such operations and consequently forces her either to communicate less information or to introduce larger (and more detectable) embedding artifacts.

A *malicious warden* can modify the exchanged images in order to impersonate Alice and trick Bob. However, this scenario requires Eve to get possession of the secret key shared between Alice and Bob. Any additional effort of Eve beyond the detection of the mere presence of the steganographic communication, for example brute-forcing the secret key and extraction of the secret message, could be attributed to the field of *forensic steganalysis*. A particularly important information may be the *length* of the hidden message. The estimation of the message length is called *quantitative steganalysis*, which outputs a non-negative real number rather than a binary decision of a classical steganalysis (the secret message is present or absent).

This dissertation focuses on developing tools for the passive warden allowing her to output a binary decision whether or not a given steganographic technique has been used.

[1] We do not elaborate on the availability of digital cameras in prison cells.

1.4 Steganography as a modern threat

Before we formalize the tasks of steganography and steganalysis and put them on a firm mathematical basis, we would like to conclude this introductory chapter by pointing out a few contemporary media articles indicating the use of modern steganography for malicious purposes.

One of the most recent cases dates to June 2010, when FBI uncovered the largest Russian spy network in the United States since the end of the cold war. According to the U.S. Department of Justice legal filings,[2] the alleged spies used steganography in digital images posted on-line as a means of communication with the Russian intelligence agency headquarters in Moscow. The news appeared in many credible media across the globe, including *The Washington Post* [101].

Apart from this officially confirmed case, government agencies keep the evidence of the actual use of steganography confidential, even though media occasionally reveal indications of such practices. Although these are more speculations than verified reports, they should not be neglected and keep digital crime investigators on alert.

Arguably, on top of the debate stands the utilization of steganography by terrorists confirmed by news articles that appear in media on a regular basis. In February 2001, *USA Today* published an article hypothesizing that Muslim extremists use steganography for planning terrorist activities [61]. The accusations reappeared in several media after the attacks of September 11, for example in *The New York Times* [83]. In 2006, Indian media reported the involvement of steganography in Mumbai July 11 train bombings [25], and in 2007 *NBC News* published a report about the steganographic practices of the Islamist terrorist group Al-Qaeda [29]. Another source pointed out an article from the Islamic magazine *Technical Mujahid* encouraging extremists to use steganography [20]. However, no direct proof of its actual use was provided.

Terrorists are not the only criminals who may employ steganographic techniques for illegal purposes. Steganography was reported to be used by South American drug dealers to communicate photographs of transit routes and cocaine shipment information.[3] According to the British *The Independent* [18], steganography was also one of the tools used by an on-line pedophile ring to distribute child pornography. The gang was caught by an international effort led by the UK's National Hi-Tech Crime Unit in 2002.

Another threat consists in the usage of steganography as a tool for an insider theft to steal company's sensitive information, such as credit card numbers or intellectual property [114]. In general, steganography may be used as a tool to conceal evidence of any criminal activity and consequently complicate the work of forensic computer analysts trying to decipher exactly what data resides on a suspect's computer hard drive [98].

The vast majority of the mentioned media articles talk about steganography in digital images because photographs are the most common steganographic carriers and the majority of available steganographic tools embed data into images [60]. We would like to note, however, that as the techniques for *detecting* steganography in digital images are maturing and as the technology advances, alternative options may come forward, for example steganography in TCP/IP streams [102] or VoIP (Voice over Internet Protocol) [97] – subliminal covert channels within on-line audio conversations.

Even though this dissertation deals solely with steganalysis of digital images, many of the ideas are general and a similar methodology may be applied to detection of steganography in other carriers as well.

1.5 Notation

Throughout the dissertation, we adhere to the following notation. A lower-case boldface letter $\mathbf{x} = (x_1, \ldots, x_n)$ stands for a vector with elements x_i, while the upper-case boldface letter $\mathbf{X} = (X_{ij})$

[2] The official complaint is available at http://www.justice.gov/opa/documents/062810complaint2.pdf.
[3] Google news report: http://afp.google.com/article/ALeqM5ieuIvbrvmfofmOt8o0YfXzbysVuQ.

represents a matrix with elements X_{ij}. The kth vector (matrix) in a sequence will be indexed with a subscript, i.e., $\mathbf{x}_k$ ($\mathbf{X}_k$). $\mathbf{I}$ is the unity matrix, and $\mathbf{X}^\mathsf{T}$ is the transpose of $\mathbf{X}$. Calligraphic font ($\mathcal{X}$) is used for sets, collections, and spaces. For a finite set $\mathcal{X}$, $|\mathcal{X}|$ denotes the number of its elements. The symbols $\mathbb{R}$ and $\mathbb{N}$ represent the real numbers and positive integers, respectively. For any $x \in \mathbb{R}$, the largest integer smaller than or equal to x is $\lfloor x \rfloor$, and the operation of rounding to an integer is denoted round(x). For any $x \in \mathbb{R}$ and $T > 0$, we define the truncation operator as

$$\text{trunc}_T(x) = \begin{cases} T \cdot \text{sign}(x) & \text{if } |x| > T, \\ x & \text{otherwise.} \end{cases} \tag{1.5.1}$$

If applied to a vector or matrix, $\text{trunc}_T(\cdot)$ is executed element-wise. Acronyms DCT and IDCT stand for Discrete Cosine Transform and its inverse, respectively, and MAD denotes median absolute deviation. The symbol P is generally reserved for the probability distribution function.

Chapter 2

Formalizing the concepts

Steganalysis tries to distinguish between cover objects and stego objects (objects containing a secret message) while steganography tries to communicate undetectably. In this chapter, we put these concepts into mathematical terms.

First, we formalize the prisoners' problem, informally introduced in Chapter 1, describe individual components of the steganographic channel and address the issue of steganographic security. Once we layout the information-theoretic foundations, we formulate the task of steganalysis as a hypothesis testing problem, and construct the optimal detector in terms of the likelihood-ratio test. Unfortunately, a practical steganalysis of digital images can hardly be implemented this way due to the incognisability of the distribution of cover images. This problem is thoroughly discussed in [11].

The inability to accurately model digital images is a major obstacle also for steganography and influences the way practical steganographic schemes are implemented. Since steganalysis needs to be aware of these trends, we devote Section 2.4 to the discussion on steganography and provide specific examples of steganographic methods that will be steganalyzed later in this dissertation.

In Section 2.5, we formulate Eve's task as a (binary) supervised classification problem and introduce feature-based steganalysis, the most common approach to practical steganalysis of digital media up to date. Afterwards, we describe the procedure for experimental evaluation of the accuracy of constructed steganalyzers.

The chapter is concluded in Section 2.7, where we provide a short historical overview of feature-based steganalysis and outline the trend towards high-dimensional image representations.

The material covered in this chapter is a well-established background rather than our contribution to the field. As the focus of this dissertation is on steganalysis, in the following text we try to keep the steganographer's perspective rather brief and refer the reader to [40] for a much more detailed discussion on steganography. We assume that the warden is passive (see Chapter 1.3) and that the cover objects are digital images.

2.1 Steganographic channel

Let $\mathcal{C}$ be the set of all cover images Alice could theoretically send to Bob. If we assume that Alice and Bob are *not* using steganography, their communication can be described by the probability distribution P_{c} defined on $\mathcal{C}$. For example, let us suppose that the images they exchange are photographs of dimensions 4000×3000 in the JPEG format, a default output format of Alice's new camera Canon PowerShot SX20 IS. In this case, even though the set $\mathcal{C}$ contains images of all possible dimensions, quality factors and theoretical scenes,[1] the probability $P_{\mathrm{c}}(\mathbf{X})$ of sending a cover image

[1] Including random scenes and other highly unlikely scenes Alice may have taken picture of.

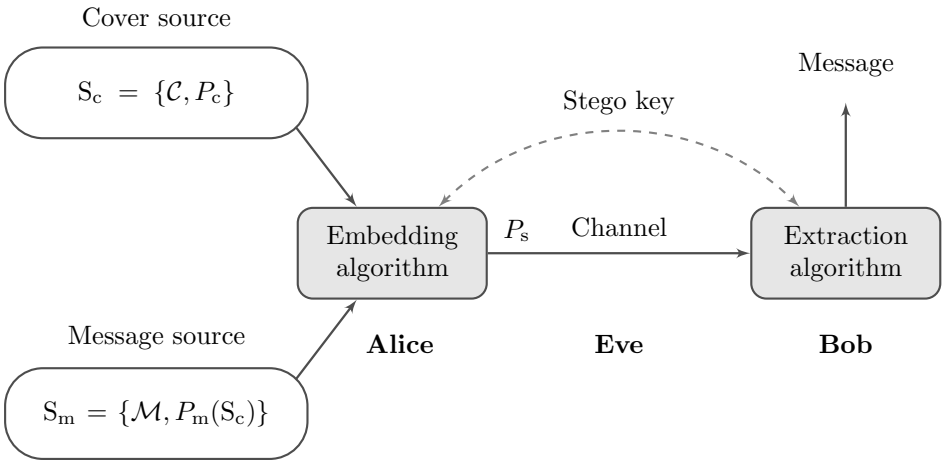


Figure 2.1.1: General form of the steganographic channel.

$\mathbf{X} \in \mathcal{C}$ that does *not* match with Alice's camera is zero. Furthermore, P_c inevitably reflects Alice's photographic habits and skills, her interests and so on. She would be unlikely to send a picture of snowy peaks if she lived in the middle of Texas, for example. We call the pair $S_c = \{\mathcal{C}, P_c\}$ the *cover source*.

Now, let us suppose that Alice and Bob use a specific steganographic tool and communicate secretly. The set of all possible messages Alice may send to Bob is denoted $\mathcal{M}$. Similarly as with the set of covers $\mathcal{C}$, we define $\mathcal{M}$ as a set of all *theoretically possible* (binary) messages of lengths $l \geq 1$. In mathematical terms,

$$\mathcal{M} = \left\{ \{0,1\}^l \, | l \in \mathbb{N} \right\}. \tag{2.1.1}$$

The secret communication between Alice and Bob can be characterized by the probability distribution P_m defined on the space of messages $\mathcal{M}$. Apart from the obvious dependence of P_m on the steganographic method Alice and Bob use, as different technique may allow them to embed more (or less) data in the same set of images, P_m is also necessarily a function of the cover source S_c as different cover objects may have different maximal allowed embedded message lengths. Alice can probably embed more information into the picture of her garden than in the image of a clear sky, as the digital representation of the former consists of a JPEG file with significantly more nonzero coefficients which are usually used for data hiding. Making the dependence of P_m on S_c explicit, we call the pair $S_m = \{\mathcal{M}, P_m(S_c)\}$ the *message source*.

In general, a steganographic algorithm consists of two components: an embedding algorithm and an extraction algorithm. The embedding algorithm, with the cover source S_c and the message source S_m at the input, produces stego images that follow a distribution P_s over $\mathcal{C}$, which is usually different from P_c. Stego images are transmitted over the channel from Alice to Bob, allowing the warden Eve to carry out her steganalysis tests. Once Bob receives a stego image, he applies the extraction algorithm and extracts the secret message. Both the embedding and extraction algorithms depend on a secret stego key shared between Alice and Bob. Figure 2.1.1 shows a schematic visualization of the steganographic channel with its individual components.

2.2 Information-theoretic foundations

The task of Alice and Bob could be formulated as follows. For a given cover source S_c, they need to create a covert communication protocol (embedding and extraction algorithms) so that the resulting stego distribution P_s is as close to the original cover distribution P_c as possible. Clearly, if P_s and

P_c were identical, no statistical test could distinguish between a legitimate communication and one contaminated by steganographic embedding.

The intuitive concept outlined in the previous paragraph was formalized within the information-theoretic framework by Cachin [19]. The closeness of two distributions can be quantified using a well-established measure called the Kullback-Leibler (KL) divergence [24] defined as

$$D_{\mathrm{KL}}(P_c||P_s) = \sum_{\mathbf{X}\in\mathcal{C}} P_c(\mathbf{X}) \log \frac{P_c(\mathbf{X})}{P_s(\mathbf{X})}. \tag{2.2.1}$$

The steganographic system is called *ϵ-secure* when $D_{\mathrm{KL}}(P_c||P_s) \leq \epsilon$. If $D_{\mathrm{KL}}(P_c||P_s) = 0$, the system is called *perfectly secure*. Taking the KL divergence (2.2.1) as a measure of steganographic security makes perfect sense as bounding $D_{\mathrm{KL}}(P_c||P_s)$ from above introduces fundamental limits on the performance of Eve's detector. To be specific, any detector Eve can construct makes two different types of errors – false alarms and missed detections. False alarms (also known as type I errors) occur when a cover image is declared as a stego image, while missed detections (type II errors) represent situations when a stego image is incorrectly classified as cover. The probability of false alarms and missed detections will be denoted as P_{FA} and P_{MD}, respectively. When Alice and Bob communicate secretly, and thus the images in the channel follow P_s, Eve's decisions follow the Bernoulli distribution with probabilities $P_s^*(0) = P_{\mathrm{MD}}$ and $P_s^*(1) = 1 - P_{\mathrm{MD}}$. Similarly, when no steganography is used, the observed images follow P_c, and Eve's detector outputs zero (cover) with probability $P_c^*(0) = 1 - P_{\mathrm{FA}}$ and one (stego) with probability $P_c^*(1) = P_{\mathrm{FA}}$. As data processing cannot increase the KL divergence, an ϵ-secure stegosystem must satisfy

$$(1 - P_{\mathrm{FA}}) \log \frac{1 - P_{\mathrm{FA}}}{P_{\mathrm{MD}}} + P_{\mathrm{FA}} \log \frac{P_{\mathrm{FA}}}{1 - P_{\mathrm{MD}}} = D_{\mathrm{KL}}(P_c^*||P_s^*) \leq D_{\mathrm{KL}}(P_c||P_s) \leq \epsilon. \tag{2.2.2}$$

For a fixed value of the detector's false alarm rate P_{FA}, this inequality imposes a lower bound on the achievable probability of missed detections, P_{MD}. For example for $P_{\mathrm{FA}} = 0$, the inequality (2.2.2) yields $P_{\mathrm{MD}} \geq e^{-\epsilon}$. Furthermore, as ϵ gets smaller (distributions P_c and P_s are becoming more similar), the lower bound on P_{MD} increases. In a limit, a perfectly secure steganographic system with $\epsilon = 0$ implies $P_{\mathrm{FA}} = 1 - P_{\mathrm{MD}}$ which corresponds to random guessing and thus the secret communication between Alice and Bob cannot be detected.

Note that the introduced concept of steganographic security was defined *with respect to* the distributions P_c and P_s. Steganographic security is therefore the property of not only the embedding algorithm, but of the whole *channel*, including the cover source $\mathrm{S_c}$ and the message source $\mathrm{S_m}$. This is in agreement with our expectations – embedding shorter messages should be more secure than embedding longer messages. Also, a perfect security of a steganographic algorithm with respect to a certain cover source does not guarantee its security with respect to a different cover source. In other words, the embedding algorithm should be always designed for a given cover source in order to maximize its security.

2.3 Steganalysis as a hypothesis-testing problem

According to the Kerckhoffs' principle, the warden Eve has a full knowledge of the steganographic channel shown in Figure 2.1.1, while the only information she does not possess is the secret stego key. In particular, she has access to the cover source $\mathrm{S_c}$ and either knows or can estimate the probability function P_c. Furthermore, she knows which steganographic algorithm Alice uses and she has access to the message source $\mathrm{S_m}$, which allows her to estimate the probability distribution P_s. Note that while assuming Eve's knowledge of the steganographic algorithm is reasonable, the knowledge of the message source might not seem so. However, this assumption is indeed indirectly followed in the vast majority of research publications on steganalysis – the steganalysis detectors are

typically constructed for a specific message length and it is commonly assumed that the messages are sequences of pseudo-random bitstreams.

From the statistical point of view, Eve's job is a simple hypothesis testing problem. She inspects the image $\mathbf{X} \in \mathcal{C}$ transmitted over the channel and needs to decide whether it follows P_{c} or P_{s}:

$$\mathrm{H}_0: \quad \mathbf{X} \sim P_{\mathrm{c}}, \tag{2.3.1}$$
$$\mathrm{H}_1: \quad \mathbf{X} \sim P_{\mathrm{s}}. \tag{2.3.2}$$

Any detector D is essentially a mapping $D : \mathcal{C} \to \{0, 1\}$, classifying an image $\mathbf{X} \in \mathcal{C}$ either as cover ($D(\mathbf{X}) = 0$) or stego ($D(\mathbf{X}) = 1$). The space $\mathcal{C}$ is thus partitioned into two parts, based on the detector's output. The set of all images $\mathbf{X} \in \mathcal{C}$ that are classified as stego images is usually called the *critical region* of the detector D and will be denoted $\mathcal{R} = \{\mathbf{X} \in \mathcal{C} | D(\mathbf{X}) = 1\}$. Eve's task is to find the best critical region $\mathcal{R}$.

In steganalysis, and in hypothesis testing in general, the importance of the two different types of errors may be different. Therefore, we introduce the cost for false alarms C_{FA} and the cost for missed detections C_{MD}. The higher is the cost C_{FA}, for example, the more important is the false alarm rate for us and therefore we would like the detector's probability of false alarms P_{FA} to be low. Under the assumption of equal prior probabilities of both hypotheses H_0 and H_1, Eve's task could then be formulated as an optimization problem

$$\min_{\mathcal{R}} \; C_{\mathrm{FA}} P(\mathbf{X} \in \mathcal{R} | \mathbf{X} \sim P_{\mathrm{c}}) + C_{\mathrm{MD}} P(\mathbf{X} \notin \mathcal{R} | \mathbf{X} \sim P_{\mathrm{s}}), \tag{2.3.3}$$

where $P(A|B)$ denotes the conditional probability of the event A, given B. The probabilities can be expressed in terms of integrals over $\mathcal{R}$ in $\mathcal{C}$ and the objective function of the optimization problem (2.3.3) reduces to

$$C_{\mathrm{FA}} \int_{\mathcal{R}} P_{\mathrm{c}}(\mathbf{X}) \mathrm{d}\mathbf{X} + C_{\mathrm{MD}} \left(1 - \int_{\mathcal{R}} P_{\mathrm{s}}(\mathbf{X}) \mathrm{d}\mathbf{X}\right) = C_{\mathrm{MD}} + \int_{\mathcal{R}} \left(C_{\mathrm{FA}} P_{\mathrm{c}}(\mathbf{X}) - C_{\mathrm{MD}} P_{\mathrm{s}}(\mathbf{X})\right) \mathrm{d}\mathbf{X}. \tag{2.3.4}$$

The above expression is minimized iff the region $\mathcal{R}$ is defined such that the integrand on the right hand side is negative which means $C_{\mathrm{FA}} P_{\mathrm{c}}(\mathbf{X}) - C_{\mathrm{MD}} P_{\mathrm{s}}(\mathbf{X}) < 0$. This is equivalent to the following condition, a standard result from statistical hypothesis testing known as the likelihood-ratio test (LRT):

$$L(\mathbf{X}) \triangleq \frac{P_{\mathrm{s}}(\mathbf{X})}{P_{\mathrm{c}}(\mathbf{X})} > \frac{C_{\mathrm{FA}}}{C_{\mathrm{MD}}} \triangleq \gamma. \tag{2.3.5}$$

We just showed that the LRT is the optimal steganalysis detector under the Bayesian setting and thus the optimal strategy of the warden Eve is to decide "stego" whenever $L(\mathbf{X}) > \gamma$. In the special case of equal costs $C_{\mathrm{FA}} = C_{\mathrm{MD}}$, this reduces to $P_{\mathrm{s}}(\mathbf{X}) > P_{\mathrm{c}}(\mathbf{X})$. It can be shown that the LRT is the optimal detector also under the Neyman-Pearson setting when the probability of false alarms is bounded from above, see, for example, Appendix D in [40].

Unfortunately, even though Eve could theoretically construct an optimal steganalyzer (steganalysis detector) as the likelihood-ratio test, this could hardly be done in practice because it requires accurate estimates of the probabilities P_{c} and P_{s} defined in $\mathcal{C}$. Going back to the example mentioned at the beginning of this chapter, suppose that Alice communicates JPEG images of dimensions 4000×3000. This makes $\mathcal{C}$ effectively a 1.2×10^7-dimensional space, and pdf estimation in such high-dimensional spaces is infeasible even if we assume that Eve has unlimited access to the cover source and uses it as an oracle.

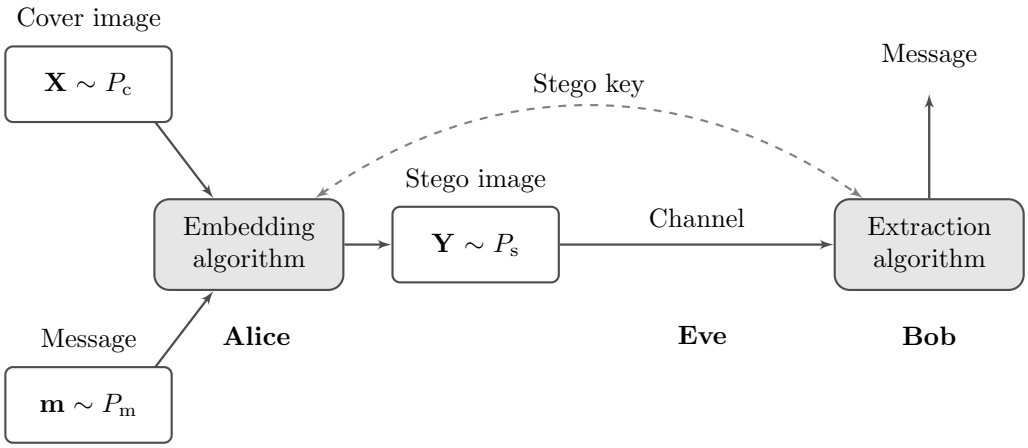


Figure 2.4.1: Steganography by cover modification – a simplified model of the steganographic channel. Compare to Figure 2.1.1

2.4 Steganography by cover modification

The inability to estimate P_{c} complicates not only steganalysis but also steganography as the goal of Alice and Bob is to design an embedding algorithm that preserves P_{c}. A rather simple way to overcome this difficulty is to embed data by *cover modification* – Alice picks a cover image $\mathbf{X} \sim P_{\mathrm{c}}$ as if there was no steganography involved, and then *modifies* it into a stego image $\mathbf{Y} \in \mathcal{C}$ in order to embed the desired message $\mathbf{m}$. The image $\mathbf{Y}$ is then communicated over the channel instead of $\mathbf{X}$. The idea is to keep the stego image $\mathbf{Y}$ in some sense "close" to $\mathbf{X}$ and hope that the resulting stego distribution P_{s} will be close to P_{c}. Even though there exist other approaches, steganography by cover modification is by far the most common embedding paradigm, at least for such a complex cover objects like digital images.

Embedding by cover modification allows Alice to hide secret messages into images "one-by-one," and the general form of the steganographic channel, as shown in Figure 2.1.1, thus simplifies. The embedding algorithm no longer accepts the whole cover source $\mathrm{S_c}$ and the message source $\mathrm{S_m}$ at the input, but already a specific cover image $\mathbf{X} \sim P_{\mathrm{c}}$ and a specific message $\mathbf{m} \sim P_{\mathrm{m}}$. See Figure 2.4.1 for a schematic visualization.

The embedding and extraction algorithms can be now represented by a pair of functions

$$Emb : \mathcal{C} \times \mathcal{M} \times \mathcal{K} \to \mathcal{C}, \tag{2.4.1}$$
$$Ext : \mathcal{C} \times \mathcal{K} \to \mathcal{M}, \tag{2.4.2}$$

where we use the symbol $\mathcal{K}$ to represent the space of all possible stego keys; typically, $\mathcal{K} \equiv \mathbb{R}$. In order to guarantee a correct message extraction, for all cover images $\mathbf{X} \in \mathcal{C}$, $\mathbf{X} \sim P_{\mathrm{c}}$, and for all possible messages $\mathbf{m} \in \mathcal{M}$ and stego keys $k \in \mathcal{K}$, the following condition needs to be satisfied:

$$Ext(Emb(\mathbf{X}, \mathbf{m}, k), k) = \mathbf{m}. \tag{2.4.3}$$

We already mentioned that the embedding function should be designed to make every stego image $\mathbf{Y} = Emb(\mathbf{X}, \mathbf{m}, k)$ as "similar" to the original cover image $\mathbf{X}$ as possible. This task has been approached from different perspectives, resulting in a range of qualitatively different embedding schemes that could be broadly divided into the following categories, each of them representing a different embedding paradigm:

- Mimic natural processing – stochastic modulation or quantization index modulation (PQ [46]), masking embedding by JPEG compression (YASS [126]),
- Preserve a *model* of cover images – statistical restoration (OutGuess [113], FCM [76]), model-based steganography (MB [116]),
- Resist existing steganalysis attacks – ± 1 embedding,
- Minimize the impact of embedding – the number of changes (F5 [132]) or a carefully designed distortion function (MME [72], HUGO [105]).

Most of the mentioned steganographic methods will be steganalyzed in this dissertation. We provide their brief description in Appendix A and refer the reader to the original publications for more details.

We note that the assignment of the steganographic methods to the individual embedding paradigms is not exclusive. For example, the ± 1 embedding, also known as LSB (Least Significant Bit) matching, evolved from simple LSB replacement in order to resist targeted attacks. At the same time, by incorporating matrix embedding, it attempts to minimize the number of changes and thus the impact of embedding. The steganographic algorithm HUGO, the most secure spatial domain algorithm up to date, works on a basis of minimizing the impact in terms of a carefully designed distortion function which relies on a complex model and thus could also be classified as a model-preserving scheme. Obviously, it also attempts to resist existing steganalysis attacks.

From now on, we will consider the simplified version of the steganographic channel as shown in Figure 2.4.1.

2.5 Steganalysis as a supervised classification

The incognisability of empirical cover sources shapes the trends of both steganography and steganalysis. In the previous section, we discussed the implications for steganography, and now we target our focus to steganalysis. As already mentioned in Section 2.3, practical steganalysis cannot be implemented as the likelihood-ratio test, at least not directly in $\mathcal{C}$. A possible way to overcome the trouble may be to work with a low-dimensional representation of images rather than with their full resolution. Suppose we construct a mapping $\mathbf{f} : \mathcal{C} \to \mathcal{F} \triangleq \mathbb{R}^d$, where d is much smaller than the dimension of the original space $\mathcal{C}$. We call the map $\mathbf{f}$ a *feature map* and the space $\mathcal{F}$ a *feature space*. Each image $\mathbf{X} \in \mathcal{C}$ is mapped to a d-dimensional vector $\mathbf{f}(\mathbf{X}) \equiv \mathbf{x} = (x_1, x_2, \dots, x_d) \in \mathcal{F}$ called a *feature vector*; its individual elements $x_i \equiv f_i(\mathbf{X})$, $i = 1, \dots, d$ are *features*. The high-dimensional distributions P_{c} and P_{s} in $\mathcal{C}$ propagate through mapping $\mathbf{f}$ into the feature space $\mathcal{F}$, where they form d-dimensional pdfs denoted as $P_{\mathrm{c}}^{(\mathcal{F})}$ and $P_{\mathrm{s}}^{(\mathcal{F})}$, respectively.

If the dimension d is very small, say $d < 5$, Eve may obtain a very accurate estimate of the distribution $P_{\mathrm{c}}^{(\mathcal{F})}$ simply by querying the cover source. Furthermore, she can obtain an estimate of $P_{\mathrm{s}}^{(\mathcal{F})}$ by applying the steganographic tool on the queried images. The Kerckhoffs' principle gives her access to both. She can now construct the LRT test and thus detect the secret communication optimally.

The problem is, however, that her optimality is *only* with respect to $\mathcal{F}$, and unless the cover/stego distinguishing properties of the original space $\mathcal{C}$ are fully carried over to the constructed feature space $\mathcal{F}$, her performance cannot be optimal with respect to $\mathcal{C}$. Unfortunately, unless the steganographic method has a serious flaw, it is unreasonable to assume that we will be able to identify such a small number of statistical quantities that would accurately distinguish between cover and stego samples. This is due to the harsh dimensionality reduction and our inability to accurately model complex dependencies among individual coefficients of natural images caused by the continuous nature of the reality and by a complex digital image acquisition process and in-camera processing. Consequently, we need more than a few features, and, as the dimension of the feature space grows,

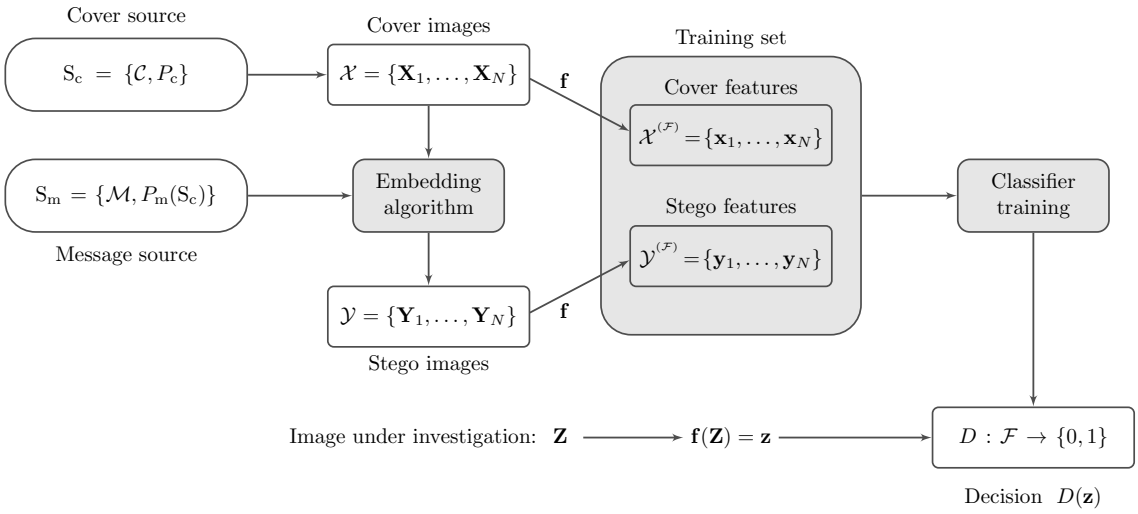


Figure 2.5.1: Steganalysis as a classification problem.

it is increasingly more difficult to estimate $P_c^{(\mathcal{F})}$ because the amount of data needed to cover the space grows exponentially with respect to dimensionality. In other words, the pdf estimation soon becomes intractable.

Therefore, steganalysis is usually approached as a supervised classification task rather than a signal-detection problem. While we still need to represent images using a reasonably low-dimensional vector of features, training a binary classifier is now significantly more manageable than estimating the probability density functions. After all, we do not *need* those pdf estimates – Eve only needs to construct a detector that would be able to distinguish between cover images and stego images. And since she has access to the cover source S_c and to the steganographic algorithm, it seems only reasonable to build a machine that could *learn* to differentiate between both classes directly from their examples.

The construction of a steganalyzer typically consists of two stages. First, Eve needs to create a feature space $\mathcal{F}$. This is an area of an active research, and finding a good feature space seems to be the Holy Grail of feature-based steganalysis. In Chapter 4, we provide some general guidelines for feature construction and, later in Chapters 5 and 6, build complex feature spaces for both spatial and DCT domain representations of images. At this point, however, we merely state that the feature space is a low-dimensional statistical descriptor of images designed to maximize the distinguishability between cover and stego images.

Once the space $\mathcal{F}$ is constructed, we need to collect a large number of cover and stego images, transform them into feature vectors, and train a binary classifier. Although there exists a large variety of machine learning tools, support vector machines [120] (SVMs) seem to be the most popular choice in steganalysis. This is due to the fact that SVMs are backed by a solid mathematical foundation cast within the statistical learning theory [130] and because they are resistant to overtraining and perform rather well even when the feature dimensionality is comparable or larger than the size of the training set. Moreover, robust and efficient open-source implementations are available for download and are easy to use [30, 21].

Figure 2.5.1 presents a conceptual diagram of feature-based steganalysis cast as a supervised classification problem. On the left hand side of the diagram is the cover source S_c and the message source S_m, both presumably accessible to the steganalyst by the Kerckhoffs' principle. Eve collects a set of cover images $\mathcal{X} = \{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N\}$ drawn according to P_c, and generates the corresponding set of stego images $\mathcal{Y} = \{\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_N\}$. The more images the better, but the limits of the classification tool and the available computational power need to be taken into account.

The next step is to map the images into the feature space $\mathcal{F}$ through the mapping $\mathbf{f}$, yielding the set of cover feature vectors $\mathcal{X}^{(\mathcal{F})} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ and stego feature vectors $\mathcal{Y}^{(\mathcal{F})} = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N\}$. Their union $\mathcal{X}^{(\mathcal{F})} \cup \mathcal{Y}^{(\mathcal{F})}$ will be used to train the classifier and is called the *training set*. Note that we always assume that the training set contains the same number of cover and stego feature vectors called cover-stego pairs. The training set is then supplied to the classifier which learns the mapping $D : \mathcal{F} \to \{0, 1\}$ (mnemonic for detector). Any new image $\mathbf{Z} \in \mathcal{C}$ passing through the channel can now be mapped into $\mathcal{F}$, $\mathbf{f}(\mathbf{Z}) = \mathbf{z}$, and classified by the detector D as $D(\mathbf{z}) \in \{0, 1\}$, zero standing for cover and 1 for stego.

2.6 Performance evaluation

In this section, we describe a procedure for evaluating the performance of the constructed classifier. This procedure is customary in research works on steganalysis and will be used in experiments within the scope of this dissertation.

The cover source $\mathrm{S_c}$ is represented by a database of cover images $\mathcal{X} = \{\mathbf{X}_1, \dots, \mathbf{X}_N\}$ collected by a steganalyst prior to the construction of the steganalyzer. Typically, the set $\mathcal{X}$ consists of images of identical sizes or JPEG quality factors in order to study the impacts of embedding systematically and to avoid misinterpretations caused by undesirable effects of other factors on the classification results. For example, the BOSSbase ver. 0.92 image database, the database of images used during the recent steganographic contest BOSS [9], consists of $9,074$ cover images taken with seven digital cameras in their RAW format, converted to grayscale, and resized/cropped to 512×512 using the script provided by the BOSS organizers. In order to assure reproducibility of steganalysis experiments, it is important to include as much information about the cover source as possible.

The next step is the generation of stego images $\mathcal{Y} = \{\mathbf{Y}_1, \dots, \mathbf{Y}_N\}$. Typically, it is assumed that the messages are pseudo-random bitstreams as the real messages could (and should) be always encrypted before embedding to enhance security. Furthermore, it is customary to fix the message length relatively to the size of the cover object – either in terms of bits per pixel (bpp) for images in raster formats or in terms of bits per non-zero AC coefficient (bpac) for JPEG images. Fixing the message length allows us to compare the security of different embedding schemes that might have a different capacity (maximum number of embeddable bits) otherwise. Furthermore, it is known that the amount of bits that could be securely transmitted over the steganographic channel is proportional to the *square root* of the cover size. In other words, Alice can embed into an image consisting of twice as many pixels only $\sqrt{2}$ times more bits at the same level of detectability. This result is known as the *square root law of steganographic capacity*, and has been confirmed both experimentally [69] and theoretically [37]. Fixing the image size eliminates the effects of the square root law on the classification results.[2] Throughout this dissertation, the relative message length will be denoted α.

After the cover images and the stego images are collected, Eve proceeds to feature extraction – she transforms all the images $\mathbf{X} \in \mathcal{X}, \mathbf{Y} \in \mathcal{Y}$ into their feature representation $\mathbf{x} \in \mathcal{X}^{(\mathcal{F})}, \mathbf{y} \in \mathcal{Y}^{(\mathcal{F})}$. The set of feature vectors $\mathcal{X}^{(\mathcal{F})} \cup \mathcal{Y}^{(\mathcal{F})}$ is then divided into two parts. One part will be used for training of the classifier and the other part will be used exclusively for the evaluation of its performance. Formally, Alice generates the set of indices $\mathcal{I}^{\mathrm{trn}} \subset \{1, 2, \dots, N\}$, $|\mathcal{I}^{\mathrm{trn}}| = N^{\mathrm{trn}}$ and forms the sets $\mathcal{X}^{\mathrm{trn}} = \{\mathbf{x}_i | i \in \mathcal{I}^{\mathrm{trn}}\}$ and $\mathcal{Y}^{\mathrm{trn}} = \{\mathbf{y}_i | i \in \mathcal{I}^{\mathrm{trn}}\}$. Their union $\mathcal{S}^{\mathrm{trn}} = \mathcal{X}^{\mathrm{trn}} \cup \mathcal{Y}^{\mathrm{trn}}$ is the set of cover-stego pairs that will be used for classifier training. The notation for the testing part is analogical: $\mathcal{I}^{\mathrm{tst}} = \{1, 2, \dots, N\} \backslash \mathcal{I}^{\mathrm{trn}}$, $\mathcal{X}^{\mathrm{tst}} = \{\mathbf{x}_i | i \in \mathcal{I}^{\mathrm{tst}}\}$, $\mathcal{Y}^{\mathrm{tst}} = \{\mathbf{y}_i | i \in \mathcal{I}^{\mathrm{tst}}\}$, and $\mathcal{S}^{\mathrm{tst}} = \mathcal{X}^{\mathrm{tst}} \cup \mathcal{Y}^{\mathrm{tst}}$. We will call the sets $\mathcal{S}^{\mathrm{trn}}$ and $\mathcal{S}^{\mathrm{tst}}$ the training and testing set, respectively. Note that in order to simplify the notation we dropped the upper index $^{(\mathcal{F})}$ when defining training and testing sets – it

[2] Fixing the relative message length in case of JPEG images with respect to non-zero AC coefficients does not alleviate the effects of the square root law as images of different content may have a very different number of non-zero coefficients which are used for data hiding. Therefore, it would be more appropriate to measure the message length in terms of the *square root* of the number of non-zero AC coefficients in the image. However, in order to stay consistent with prior art, we stick to the established measure.

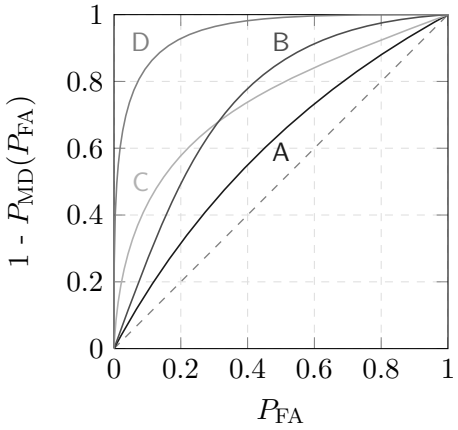


Figure 2.6.1: Examples of ROC curves. Curve A represents a bad detector, while D represents a fairly good detector. Curves B and C illustrate the difficulty of comparing ROC curves as they intersect.

is implicitly assumed that the classifier operates in the feature space $\mathcal{F}$ rather than in the original space of images $\mathcal{C}$. Having said this, we may sometimes slightly abuse the notation regarding the term *training set* as we may mean either the set of *images* or the set of *features* used for training, depending on the context. Furthermore, in Figure 2.5.1, we called the training set the set of *all* cover-stego pairs available to Eve, however, here we use the same name only for a portion explicitly defined by the index set $\mathcal{I}^{\mathrm{trn}}$. To avoid the confusion, intuitively, by training set we always mean the set of samples the classifier is being *trained* on. In this case, we distinguish it from the testing set $\mathcal{S}^{\mathrm{tst}}$ whose purpose is to "simulate" the steganographic channel and to provide us with an estimate of a real performance.

We would like to point out that the testing set $\mathcal{S}^{\mathrm{tst}}$ does not contain any stego image whose cover would be in the training part $\mathcal{S}^{\mathrm{trn}}$ and vice versa. In other words, the cover-stego pairs are preserved during the process of dividing the original database into training and testing parts. This makes intuitively sense as the constructed classifier should have zero information about the images in the testing set, apart from the fact that they come from the same cover source. We refer the reader to Appendix C.3 for more information about cover-stego pairs in steganalysis and their impact on classification.

The resulting detector $D : \mathcal{F} \to \{0, 1\}$ is fully determined by the training set $\mathcal{S}^{\mathrm{trn}}$ and the chosen classification tool. The testing phase then consists of applying D to all the testing samples from $\mathcal{S}^{\mathrm{tst}}$ and assessing the estimates of the probability of false alarms and missed detections:

$$P_{\mathrm{FA}} = \frac{1}{|\mathcal{X}^{\mathrm{tst}}|} \left|\{\mathbf{x} | \mathbf{x} \in \mathcal{X}^{\mathrm{tst}} \text{ and } D(\mathbf{x}) = 1\}\right|, \tag{2.6.1}$$

$$P_{\mathrm{MD}} = \frac{1}{|\mathcal{Y}^{\mathrm{tst}}|} \left|\{\mathbf{y} | \mathbf{y} \in \mathcal{Y}^{\mathrm{tst}} \text{ and } D(\mathbf{y}) = 0\}\right|. \tag{2.6.2}$$

For a given detector D, the curve $1 - P_{\mathrm{MD}}(P_{\mathrm{FA}})$ is called the Receiver-Operating-Characteristic (ROC) curve and describes the detector's performance. It is usually possible to adjust the final threshold of the classifier and thus obtain the whole ROC curve defined for $P_{\mathrm{FA}} \in [0, 1]$. A few example ROC curves are shown in Figure 2.6.1. Since the diagonal corresponds to a random guessing detector, the ROC curves close to the diagonal (curve A) thus correspond to poor steganalyzers. On the other hand, the point $[0, 1]$ corresponds to perfect detectability as $P_{\mathrm{FA}} = P_{\mathrm{MD}} = 0$, and the ROC curves close to this point are thus those of good steganalysis methods (curve D). ROC curves may be hard to compare as they may intersect and thus one detector may be better than the other one only for certain false alarms (curves B and D). Therefore, a scalar quantity is usually

calculated from the ROC curve as these can be unambiguously ordered. While there exist several such measures proposed in the literature, we will exclusively use the minimal average error under equal prior probabilities:

$$P_{\mathrm{E}} = \min_{P_{\mathrm{FA}}} \frac{P_{\mathrm{FA}} + P_{\mathrm{MD}}(P_{\mathrm{FA}})}{2}, \tag{2.6.3}$$

which is the measure that was previously used in [126, 51, 69]. The error P_{E} lies in the range $[0, 0.5]$, zero corresponding to perfect detection and 0.5 to random guessing (and thus perfect security of the steganographic scheme).

2.7 Past, present, and future of feature-based steganalysis

Let us conclude this chapter by a short historical narrative of machine-learning based steganalysis conducted in feature spaces, while trying to anticipate the future development.

To the best of our knowledge, the first use of machine learning in the field is due Avcıbaş *et al.* [7] where the authors used a 18-dimensional feature vector consisting of binary similarity measures, closely followed by Farid *et al.* [31] where the authors used 72 higher-order moments of coefficients obtained by transforming an image using quadratic mirror filters as features.[3] Both works utilized SVMs for classification. Generally speaking, early feature-based steganalysis algorithms used only a few dozens of features, e.g., 18 binary similarity metrics [5], 23 DCT features [39], and 27 higher-order moments of wavelet coefficients [52]. However, increased sophistication of steganographic algorithms together with the desire to detect steganography more accurately prompted steganalysts to use feature vectors of increasingly higher dimension. The feature set described in [107] used 274 features and was later extended to twice its size [77] by Cartesian calibration,[4] while 324- and 486-dimensional feature vectors were proposed in [123] and [22], respectively. In [96], the authors used 432-dimensional feature vector formed from higher-order magnitude and phase statistics extracted in the wavelet domain, and the SPAM set for the second-order Markov model of pixel differences has a dimensionality of 686 [104]. Additionally, it proved beneficial to merge features computed from different domains to further increase diversity. The 1234-dimensional Cross-Domain Feature set (CDF) [82] proved especially effective against YASS [126, 125] (see Appendix A.8), which makes embedding changes in a key-dependent domain. Because modern steganography [105, 32, 94] places embedding changes in those regions of images that are hard to model, increasingly more complex statistical descriptors of covers are required to capture a larger number of (weaker) dependencies among cover elements that might be disturbed by embedding [50, 49, 54, 78].

This short overview clearly underlies the high-dimensional trend of feature space design, which additionally necessitates using larger training sets of images. The complexity of SVM training, however, slows down the development cycle even for problems of a moderate size as the complexity of calculating the Gram matrix representing the kernel is proportional to the square of the product of the feature dimensionality d and the training set size ($2N$). Moreover, the training itself is at least quadratic in N. This imposes limits on the size of the problem one can handle in practice and forces the steganalyst to consciously design the features to fit within the complexity constraints defined by available computing resources.

We believe that machine learning should *not* be an obstacle in the feature space design, and, in order to facilitate further development of steganalysis, a scalable classification tool is needed. For this purpose, we developed an ensemble classification framework built as a combination of simple base learners that are inexpensive to train. We will show that the proposed ensemble system is capable of training on large training sets and handling feature spaces of very high dimensions (tens of thousands), allowing us to re-think the way feature spaces have been constructed so far.

[3] This technique was later extended to color images by combining all three color channels [95].
[4] The concept of Cartesian calibration will be addressed in Chapter 4.2.

Chapter 3

Ensemble classifier

As modern steganalysis relies on increasingly more complex image models, there is a growing need for a scalable machine learning tool. In order to address the complexity issues and to facilitate future development in steganalysis, we propose to use ensemble classifiers – a classification framework built as a fusion of decisions of inexpensively trained base learners.

Our first promising results in this direction were published in [78], followed by [81] where we introduced a fully automatized framework. We utilized the ensemble classifier during the recent steganalysis competition BOSS [9] and continued to pursue this exciting direction, giving birth to a series of publications [50, 49, 48, 79].

In this chapter, we fully describe the ensemble classification framework and its implementation. We start by introducing the basic algorithm, illustrate its dependence on two hyper-parameters, and complete the description in Section 3.2 where we describe an algorithm for automatic determination of these parameters.

Sections 3.3 and 3.4 discuss various implementation issues and the relationship of the proposed system to prior art, highlighting its similarity to random forests proposed by Breiman [14]. In Sections 3.5 and 3.6, we discuss and experimentally compare two alternative designs of the framework that incorporate the techniques of cross-validation and AdaBoost [38], respectively.

Finally, in Section 3.7, the performance of the proposed ensemble framework is compared with the performance of SVMs under several different steganalysis scenarios in terms of both the training complexity and the achieved detection accuracy. The chapter is summarized in Section 3.8.

3.1 Algorithm

The proposed ensemble classifier consists of many independently trained base learners – simple classifiers built on (uniformly) randomly selected subspaces of the original feature space $\mathcal{F}$. The dimensionality of the random subspaces can be chosen to be much smaller than the full dimensionality d, which significantly decreases the training complexity. Given an example from the testing set, the final decision is formed by aggregating the decisions of individual base learners. Even though the performance of individual base learners can be weak, the accuracy quickly improves after fusion and eventually levels out for a sufficiently large number of base learners. This supervised ensemble strategy will work only if the individual base learners are sufficiently diverse in the sense that they make different errors on unseen data. In order to increase the mutual diversity of base learners, each of them is trained on a bootstrap sample[1] drawn from the training set rather than on the whole training set. This strategy, known in the machine learning community as bootstrap aggregating or

[1] Bootstrap sample is a uniform sample with replacement.

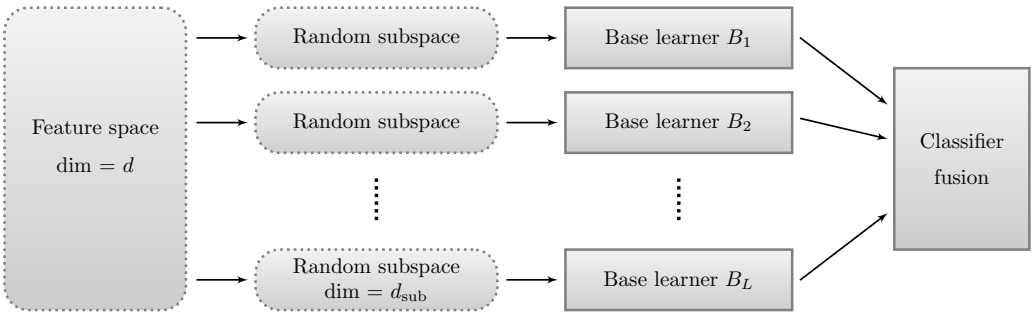


Figure 3.1.1: Diagram illustrating the proposed ensemble classifier. The random subspaces are constructed by selecting $d_{\text{sub}} \ll d$ features randomly and uniformly from the entire feature space $\mathcal{F}$.

bagging [13], will also allow us to obtain an accurate estimate of the testing error, which will be important for determining optimal ensemble parameters.

To formally describe our ensemble classifier, we adhere to the notation introduced in the previous chapter. In particular, N is the total number of available cover-stego pairs, $\mathcal{I}^{\text{trn}}$ and $\mathcal{I}^{\text{tst}}$ are sets of indices determining the training set $\mathcal{S}^{\text{trn}} = \{\mathbf{x}_i, \mathbf{y}_i | i \in \mathcal{I}^{\text{trn}}\}$ and the testing set $\mathcal{S}^{\text{tst}} = \{\mathbf{x}_i, \mathbf{y}_i | i \in \mathcal{I}^{\text{tst}}\}$, respectively, with the letter $\mathbf{x}$ standing for cover and $\mathbf{y}$ for stego feature vectors. $N^{\text{trn}} = |\mathcal{I}^{\text{trn}}|$ and $N^{\text{tst}} = |\mathcal{I}^{\text{tst}}|$, satisfying $N = N^{\text{trn}} + N^{\text{tst}}$, are the numbers of training and testing cover-stego pairs. Additionally, we reserve d_{sub} for the dimensionality of the subspace of $\mathcal{F}$ on which each base learner operates and L is the number of base learners. The lth random subspace is defined by the index set $\mathcal{D}_l \subset \{1, \dots, d\}$, $|\mathcal{D}_l| = d_{\text{sub}}$, and the symbol $\mathbf{x}^{(\mathcal{D}_l)}$ is a feature vector consisting only of those features from $\mathbf{x}$ whose indices are in $\mathcal{D}_l$, preserving their original order.

The individual base learners B_l, $l = 1, \dots, L$, are mappings $\mathbb{R}^{d_{\text{sub}}} \to \{0, 1\}$, where 0 stands for cover and 1 for stego. The lth base learner is built on the training set

$$\mathcal{S}_l^{\text{trn}} = \left\{ \mathbf{x}_m^{(\mathcal{D}_l)}, \mathbf{y}_m^{(\mathcal{D}_l)} \right\}_{m \in \mathcal{N}_l^{\text{b}}}, \tag{3.1.1}$$

where the index set $\mathcal{N}_l^{\text{b}}$ is the lth bootstrap sample drawn from the set $\mathcal{I}^{\text{trn}}$, $|\mathcal{N}_l^{\text{b}}| = N^{\text{trn}}$. Note that we assume that bootstrap samples are formed to preserve cover-stego pairs. This steganalysis-specific decision is rather important as it has been shown that breaking cover-stego pairs into two sets, one of which is used for training and the other one for error estimation, may lead to a biased error estimate and, consequently, to a suboptimal performance [121, 74]. We discuss this topic later in Appendix C.3. The decision threshold of each base learner is adjusted to minimize the total detection error under equal priors on the training set, P_{E}, defined by equation (2.6.3).

We recommend to implement each base learner as the Fisher Linear Discriminant (FLD) [27] because of its low training complexity; the most time consuming part is forming the within-class covariance matrices and inverting their summation. Additionally, such weak and unstable classifiers desirably increase diversity. Since the FLD is a standard classification tool, we only describe those parts of it that are relevant for the ensemble classifier. The lth base learner is fully described using the generalized eigenvector

$$\mathbf{v}_l = (\mathbf{S}_{\text{W}} + \lambda \mathbf{I})^{-1} (\boldsymbol{\mu}_{\mathbf{x}} - \boldsymbol{\mu}_{\mathbf{y}}), \tag{3.1.2}$$

where $\boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\mu}_{\mathbf{y}} \in \mathbb{R}^{d_{\text{sub}}}$ are the means of each class

$$\boldsymbol{\mu}_{\mathbf{x}} = \frac{1}{N^{\text{trn}}} \sum_{m \in \mathcal{N}_l^{\text{b}}} \mathbf{x}_m^{(\mathcal{D}_l)}, \quad \boldsymbol{\mu}_{\mathbf{y}} = \frac{1}{N^{\text{trn}}} \sum_{m \in \mathcal{N}_l^{\text{b}}} \mathbf{y}_m^{(\mathcal{D}_l)}, \tag{3.1.3}$$

Algorithm 3.1 Ensemble classifier, parametrized by d_{sub} and L.

1: **for** l=1 to L **do**

2: Form a random subspace

$$\mathcal{D}_l \subset \{1, \ldots, d\}, |\mathcal{D}_l| = d_{\text{sub}} < d$$

3: Form a bootstrap sample $\mathcal{N}_l^{\text{b}}, \left|\mathcal{N}_l^{\text{b}}\right| = N^{\text{trn}}$ by uniform sampling with replacement from $\mathcal{I}^{\text{trn}}$

4: Train a base learner B_l on features

$$\mathcal{S}_l^{\text{trn}} = \left\{\mathbf{x}_m^{(\mathcal{D}_l)}, \mathbf{y}_m^{(\mathcal{D}_l)}\right\}_{m \in \mathcal{N}_l^{\text{b}}}$$

$\Rightarrow$ obtain eigenvector $\mathbf{v}_l$ and threshold T_l

5: For all test examples $\mathbf{z} \in \mathcal{S}^{\text{tst}}$ make lth decision:

$$B_l(\mathbf{z}^{(\mathcal{D}_l)}) \triangleq \begin{cases} 1 & \text{when } \mathbf{v}_l^{\mathsf{T}} \mathbf{z}^{(\mathcal{D}_l)} > T_l \\ 0 & \text{otherwise.} \end{cases}$$

6: **end for**

7: Form the final decisions $B(\mathbf{z})$ by majority voting:

$$B(\mathbf{z}) = \begin{cases} 1 & \text{when } \sum_{l=1}^{L} B_l(\mathbf{z}^{(\mathcal{D}_l)}) > L/2 \\ 0 & \text{when } \sum_{l=1}^{L} B_l(\mathbf{z}^{(\mathcal{D}_l)}) < L/2 \\ \text{random} & \text{otherwise.} \end{cases}$$

8: **return** $B(\mathbf{z}), \mathbf{z} \in \mathcal{S}^{\text{tst}}$

$$\mathbf{S}_{\text{W}} = \sum_{m \in \mathcal{N}_l^{\text{b}}} (\mathbf{x}_m^{(\mathcal{D}_l)} - \boldsymbol{\mu}_{\mathbf{x}})(\mathbf{x}_m^{(\mathcal{D}_l)} - \boldsymbol{\mu}_{\mathbf{x}})^{\mathsf{T}} + (\mathbf{y}_m^{(\mathcal{D}_l)} - \boldsymbol{\mu}_{\mathbf{y}})(\mathbf{y}_m^{(\mathcal{D}_l)} - \boldsymbol{\mu}_{\mathbf{y}})^{\mathsf{T}} \tag{3.1.4}$$

is the within-class scatter matrix, and λ is a stabilizing parameter to make the matrix $\mathbf{S}_{\text{W}} + \lambda \mathbf{I}$ positive definite and thus avoid problems with numerical instability in practice when $\mathbf{S}_{\text{W}}$ is singular or ill-conditioned.[2]

For a test feature vector $\mathbf{z} \in \mathcal{S}^{\text{tst}}$, the lth base learner reaches its decision by computing the projection $\mathbf{v}_l^{\mathsf{T}} \mathbf{z}^{(\mathcal{D}_l)}$ and comparing it to a threshold (previously adjusted to meet a desired performance criterion). After collecting all L decisions, the final classifier output is formed by combining them using an unweighted (majority) voting strategy – the sum of the individual votes is compared to the decision threshold $L/2$. We note that this threshold may be adjusted within the interval $[0, L]$ in order to control the importance of the two different types of errors or to obtain the whole receiver operating characteristic (ROC curve). In all experiments in this paper, we adjust the threshold to $L/2$ as P_{E} (2.6.3) is nowadays considered standard for evaluating the accuracy of steganalyzers in practice.

The pseudo-code for the entire ensemble classifier is described in Algorithm 3.1, while Figure 3.1.1 shows its high-level conceptual diagram. The classifier depends on two parameters, d_{sub} and L, which are determined using algorithms from Section 3.2.

3.1.1 Illustrative example

Before finishing the description of the ensemble classifier with procedures for automatic determination of d_{sub} and L, we include a simple illustrative example to demonstrate the effect of the

[2] The parameter λ can be either fixed to a small constant value (e.g., $\lambda = 10^{-10}$) or dynamically increased once numerical instability is detected.

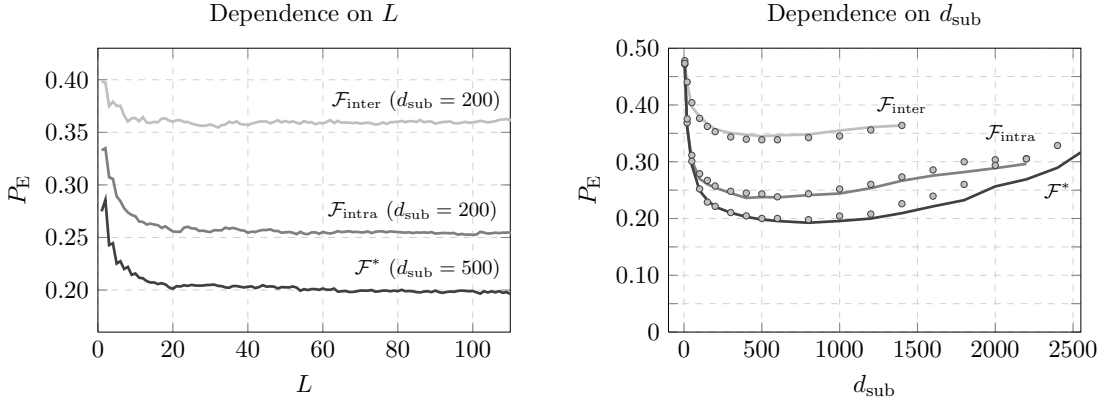


Figure 3.1.2: Left: Detection error P_E quickly saturates with the number of fused learners L. Right: The detection error after saturation as a function of d_{sub}. The dots represent out-of-bag error estimates E_{OOB} (see Section 3.2). Feature sets considered: $\mathcal{F}_{inter}$, $\mathcal{F}_{intra}$, and $\mathcal{F}^*$ with dimensionalities 1550, 2375, and 3925 (see [81]). Target algorithm: nsF5 with payload 0.1 bpac.

parameters on performance. We do so for the steganographic algorithm nsF5 (no-shrinkage F5) [51] (see Appendix A.5) as a modern representative of steganographic schemes for the JPEG domain, using a simulation of its embedding impact if optimal wet-paper codes were used.[3]

The cover source is formed by the CAMERA database containing 6,500 JPEG images and described in more details in Appendix B. The images were randomly divided into two halves for training and testing, respectively ($N^{trn} = N^{tst} = N/2$).

All ensemble classifiers were built to detect stego images embedded with relative payload 0.1 bpac (bits per non-zero AC DCT coefficient). Figure 3.1.2 (left) shows the classifier error P_E (2.6.3) on the testing set $\mathcal{S}^{tst}$ as a function of the number of fused base learners L, for three different feature sets and a fixed d_{sub}. The feature sets $\mathcal{F}_{inter}$, $\mathcal{F}_{intra}$, and $\mathcal{F}^*$ are properly defined in [81]; here we use them merely to illustrate that the classification accuracy quickly saturates with L.[4]

The error P_E (after saturation) is shown as a function of the subspace dimensionality d_{sub} in Figure 3.1.2 (right). Observe an initial quick drop followed by a fairly flat minimum, after which P_E starts growing again, which is mainly because of the following two reasons. First, the individual base learners become more dependent and thus the ability of the ensemble classifier to form non-linear boundaries decreases. Second, the individual FLDs start to suffer from overtraining as the subspace dimensionality increases while the training set size remains the same.

3.2 Parameter determination

To complete the description of the classifier, we now supply a procedure for determining d_{sub} and L. Since each base learner B_l is trained only on a bootstrap sample $\mathcal{S}_l^{trn}$ of the full training set $\mathcal{S}^{trn}$ (see definition (3.1.1)), roughly 37% of cover-stego pairs are *not* used for its training and can be used for validation as B_l provides them with a single vote.[5] Therefore, after n base learners are trained, each training sample $\mathbf{z} \in \mathcal{S}^{trn}$ will collect on average $0.37n$ predictions that can be fused using the majority voting strategy into a prediction $B^{(n)}(\mathbf{z}) \in \{0, 1\}$. The following is an unbiased

[3] A simulator of nsF5 embedding is provided at http://dde.binghamton.edu/download/nsf5simulator/.

[4] Broadly speaking, $\mathcal{F}_{inter}$ and $\mathcal{F}_{intra}$ capture spatial (inter-block) and frequency (intra-block) dependencies among DCT coefficients and $\mathcal{F}^*$ is their union.

[5] This procedure is an alternative to cross-validation in SVM training.

estimate of the testing error known as the "out-of-bag" (OOB) estimate:

$$E_{\text{OOB}}^{(n)} = \frac{1}{2N^{\text{trn}}} \sum_{m \in \mathcal{I}^{\text{trn}}} \left(B^{(n)}(\mathbf{x}_m) + 1 - B^{(n)}(\mathbf{y}_m) \right). \tag{3.2.1}$$

The term comes from bagging (bootstrap aggregating) which is a well-established technique for reducing the variance of classifiers [13]. In contrast to bagging, we use a different random subset of features for training each base learner. Figure 3.1.2 (right) illustrates that $E_{\text{OOB}}^{(n)}$ is indeed an accurate estimate of the testing error.

3.2.1 Stopping criterion for L

As is already apparent from Figure 3.1.2 (left), the classification accuracy saturates rather quickly with L. The speed of saturation, however, depends on the accuracy of individual base learners, on the relationship between d and d_{sub}, and is also data dependent. Therefore, we determine L dynamically by observing the progress of the OOB estimate (3.2.1) and stop the training once the last K moving averages calculated from μ consecutive E_{OOB} values lie in an ϵ-tube:

$$L = \arg\min_n \left\{ n; \left| \min_{i \in L_K(n)} M_\mu(i) - \max_{i \in L_K(n)} M_\mu(i) \right| < \epsilon \right\}, \tag{3.2.2}$$

where

$$M_\mu(i) = \frac{1}{\mu} \sum_{j=i-\mu+1}^{i} E_{\text{OOB}}^{(j)} \tag{3.2.3}$$

and $L_K(n) = \{n-K, \ldots, n\}$. The parameters K, μ, and ϵ are user-defined and control the trade-off between computational complexity and optimality. In all our experiments, we fixed $K = 50$, $\mu = 5$, and $\epsilon = 0.005$ and observed that this choice of the parameters works universally for different steganalysis tasks, i.e., for both small and large values of d and d_{sub}, for a wide range of payloads (low and high E_{OOB} values), and different steganographic algorithms.

Note that while $E_{\text{OOB}}^{(L)}$ is formed by fusing $0.37L$ decisions (on average), the predictions on testing samples will be formed by fusing all L decisions. The final out-of-bag estimate meeting the criterion (3.2.2) will be denoted $E_{\text{OOB}} \equiv E_{\text{OOB}}^{(L)}$.

3.2.2 Subspace dimension d_{sub}

Since the classification accuracy is fairly insensitive to d_{sub} around its minimum (see Figure 3.1.2 (right)), most simple "educated guesses" of d_{sub} give already near-optimal performance, which is important for obtaining quick insights for the analyst. Having said this, we now supply a formal procedure for automatic determination of d_{sub}. Because $P_{\text{E}}(d_{\text{sub}})$ is unimodal in d_{sub}, the minimum can be found through a one-dimensional search over d_{sub} using $E_{\text{OOB}}(d_{\text{sub}})$ as an estimate of $P_{\text{E}}(d_{\text{sub}})$. Since the matrix inversion in (3.1.2) requires $O(d_{\text{sub}}^3)$ operations, to avoid evaluating $E_{\text{OOB}}(d_{\text{sub}})$ for large values of d_{sub}, we approach the minimum "from the left" using a simple direct-search derivative-free technique inspired by the compass search [84]. The pseudo-code, shown in Algorithm 3.2, can be interpreted as follows. Starting with a small value of d_{sub}, we keep increasing it by a pre-defined step Δ_d as long as $E_{\text{OOB}}(d_{\text{sub}})$ decreases. Once the error passes its minimum and starts increasing again, we go back to the lowest point and the step size is refined, $\Delta_d \leftarrow \Delta_d/2$, until the solution is found within the desired tolerance τ (Stage 2).

The parameters τ, Δ_d, and ϵ_d control the trade-off between the training time and optimality of the solution. In particular, τ specifies the desired relative tolerance within which the lowest value of E_{OOB} is to be found, Δ_d determines the initial step size, and ϵ_d specifies the robustness w.r.t.

Algorithm 3.2 One-dimensional search for d_{sub}. To simplify the boundary issues, we define $E_{\text{OOB}}(d_{\text{sub}}) = 1$ for all $d_{\text{sub}} \notin [0, d]$.

Set parameters τ, Δ_d, and ϵ_d
//Stage 1: first pass with Δ_d
Initialize $k \leftarrow 0$, $E^*_{\text{OOB}} \leftarrow 1$, $d^*_{\text{sub}} \leftarrow 0$
repeat
  $k \leftarrow k + 1$
  Train ensemble classifier and obtain out-of-bag error estimate $E_{\text{OOB}}(k\Delta_d)$
  if $E_{\text{OOB}}(k\Delta_d) < E^*_{\text{OOB}}$ **then**
    $E^*_{\text{OOB}} \leftarrow E_{\text{OOB}}(k\Delta_d)$
    $d^*_{\text{sub}} \leftarrow k\Delta_d$
  end if
until $E_{\text{OOB}}(k\Delta_d) > E^*_{\text{OOB}} + \epsilon_d$
//Stage 2: localize the minimum by refining Δ_d
repeat
  Obtain $E_1 \equiv E_{\text{OOB}}(d^*_{\text{sub}} - \Delta_d)$
  Obtain $E_2 \equiv E_{\text{OOB}}(d^*_{\text{sub}})$
  Obtain $E_3 \equiv E_{\text{OOB}}(d^*_{\text{sub}} + \Delta_d)$
  if $1 \geq \frac{2E_2}{E_1+E_3} > 1 - \tau$ or Δ_d too small **then**
    return d^*_{sub}
  else
    $E^*_{\text{OOB}} \leftarrow \min\{E_1, E_2, E_3\}$
    $d^*_{\text{sub}} \leftarrow d_{\text{sub}}$ yielding E^*_{OOB}
    $\Delta_d \leftarrow \Delta_d/2$
  end if
until 1

statistical fluctuations of $E_{\text{OOB}}(d_{\text{sub}})$ – it identifies the moment when to stop increasing d_{sub} and proceed to Stage 2.

Similarly as with the parameters K, μ, and ϵ for the determination of the number of base learners L, the choice of the parameters τ, Δ_d, and ϵ_d seems to be rather universal – in all experiments conducted in this dissertation we used $\tau = 0.02$, $\Delta_d = 200$, and $\epsilon_d = 0.005$ as a good compromise between the training time and the classification accuracy.

3.3 Relationship to prior art

Boosting [119] is a general method of creating an accurate predictor by combining many weaker learners through a properly chosen aggregation strategy. Since the first successful ensemble systems were proposed, boosting has evolved into a well developed discipline whose popularity keeps on growing due to the simplicity of the approach, its straightforward parallel implementation, and high accuracy.

One of the earliest boosting frameworks is AdaBoost proposed by Freund and Schapire [38]. AdaBoost trains individual weak learners (base learners) sequentially and every base learner focuses on those samples that were more difficult to classify by previous base learners. This is achieved by

a continuous adjustment of the training sample weights throughout the learning – the weights of training samples that were misclassified are increased while the weights of those that were classified correctly are decreased. The final decision is formed as a weighted combination of individual predictions with weights corresponding to the standalone accuracy of each base learner. AdaBoost is a deterministic meta-algorithm applicable to any classifier (base learner) capable of handling weighted training samples.

A different way of boosting the performance through an ensemble of weaker learners is bagging (or bootstrap aggregating) due Breiman [13], a concept already mentioned in Section 3.1 as it is part of our proposed steganalysis framework. In bagging, every base learner is trained on a different bootstrap sample drawn from the original training set and their individual predictions are then combined through a simple majority voting scheme (averaging). The success of bagging relies on the *instability* of base learners w.r.t. small changes in the training set. An important by-product of bagging is the ability to continuously monitor the testing error estimate (OOB).

The random forest [14] is an extended version of bagging in the sense that it also trains individual base learners on bootstrap samples of the training set. The base learners are, however, additionally randomized by making them dependent on a random vector that is drawn independently and from a fixed distribution. In [14], each base learner is a decision tree whose splitting variables are chosen randomly as a small subset of the original variables. The final prediction is again formed as a majority vote. This additional randomization introduces instability (and thus diversity) to the individual base learners and substantially speeds-up the training. On the other hand, the accuracy of individual base learners decreases, which is to be expected. However, it turns out that the combined prediction generally yields comparable or even better results than bagging or AdaBoost. We would like to stress that unlike in AdaBoost, the random forest treats individual base learners equally in forming the final decision – this is because all the base learners were generated using the same random procedure.

Our steganalysis system proposed in Section 3.1 could be categorized as a random forest with the FLD as a base learner. The random component is in the feature subspace generation and is a crucial part of the system as using the full feature space would be computationally intractable due to high dimensionality.

The idea of forming random subspaces from the original feature space is not new and is known under different names. Decision forests [58], attribute bagging [17], CERP (Classification by Ensembles from Random Partitions) [1], or the recently proposed RSE (Random Subsample Ensemble) [122] are all ensemble-based classifiers sampling the feature space prior base learner training to either increase the diversity among classifiers or reduce the original high dimension to manageable values.

Most ensemble systems described in the literature use base learners implemented as classification trees even though other classifiers may be used. For example, SVMs are used as base learners in [86], while in [3] a set of different base learners are compared, including logistic regression, linear SVM, and FLD. Our decision to select the FLD was based on numerous experiments we performed and will be discussed in more detail in the next section. Briefly, FLDs are very fast and provided overall good performance when combined together into a final vote.

Besides ensemble classification, there exist numerous other well-developed strategies for reducing the training complexity. One popular choice are dimensionality reduction techniques that can be either unsupervised (PCA) or supervised (e.g., feature selection [87]) applied prior to classification as a part of the feature pre-processing. However, such methods are rarely suitable for applications in steganalysis when no small subset of features can deliver performance similar to the full-dimensional case. The dimensionality reduction and classification can be performed simultaneously either by minimizing an appropriately constructed single objective function directly (SVDM [103]) or by constructing an iterative algorithm for dimensionality reduction with a classification feedback after every iteration. In machine learning, these methods are known as embedded and wrapper methods [87].

Finally, the idea of using a committee of detectors *for steganalysis* appeared in [70]. However, the focus of the work was elsewhere – several classifiers were trained individually to detect different steganographic methods and their fusion was shown to outperform a single classifier trained on a mixture of stego images created by those methods.

3.4 Discussion

To the best of our knowledge, a fully automatized framework combining random feature subspaces and bagging into a random forest classifier, together with an efficient utilization of out-of-bag error estimates for stopping criterion and the search for the optimal value of the subspace dimension is a novel contribution not only in the field of steganalysis, but also in the ensemble classification literature. The ensemble classifier as described in Section 3.1 provided the best overall performance and complexity among many different versions we have investigated. In particular, we studied whether it is possible to improve the performance by selecting the features randomly but non-uniformly and we tested other base learners and aggregation rules for the decision fusion. Even though none of these modifications brought an improvement, we believe that they deserve to be commented on and we discuss them in this section.

Depending on the feature set and the steganographic algorithm, certain features react more sensitively to embedding than others. Thus, it seemingly makes sense to try improve the performance by selecting the more influential features more frequently instead of uniformly at random. We tested biasing the random selection to features with a higher individual Fisher ratio. However, any deviation from the uniform distribution lead to a drop in the performance of the entire ensemble. This is likely because biased selection of features decreases the diversity of the individual base learners. The problem of optimum trade-off between diversity and accuracy of the base learners is not completely resolved in the ensemble literature [99, 16] and we refrain from further analyzing this important issue in this dissertation.

Next, we investigated whether base learners other than FLDs can improve the performance. In particular, we tested linear SVMs (L-SVMs), kernelized FLDs [100], decision trees, naive Bayesian classifiers, and logistic regression. In summary, none of these choices proved a viable alternative to the FLD. Decision trees were unsuitable due to the fact that in steganalysis it is unlikely to find a small set of influential features (unless the steganography has some fundamental weakness). All features are rather weak and only provide detection when considered as a whole or in large subsets. Interestingly, the ensemble with kernelized FLD, L-SVM, and logistic regression had performance comparable to FLDs, even though the individual accuracies of base learners were higher. Additionally, the training complexity of these alternative base learners was much higher. Also, unlike FLD, both L-SVM and kernelized FLD require pre-scaling of features and a parameter search, which further increases the training time.

The last element we tested was the aggregation rule. The voting as described in Algorithm 3.1 could be replaced by more sophisticated rules [85]. For example, when the decision boundary is a hyperplane, one can compute the projections of the test feature vector on the normal vector of each base learner and threshold their sum over all base learners. Alternatively, one could take the sum of log-likelihoods of each projection after fitting models to the projections of cover and stego training features (the projections are well-modeled by a Guassian distribution). We observed, however, that all three fusion strategies gave essentially identical results. Thus, we selected the simplest rule – the majority voting as our final choice.

Apart from optimizing individual components of the system, we also tried two alternative designs of the framework as a whole. First, we replaced bootstrapping and out-of-bag error estimation with a k-fold cross-validation. The second direction of our efforts was to incorporate the ideas of AdaBoost and to assign weights to the training samples and/or base learners. Both ideas are natural to try and thus we implemented them and subjected to comparative tests. Even though none of them resulted in performance improvement, we believe it is valuable to expose our investigative experiments to the community and thus decided to provide more details in the next two sections.

3.5 Cross-validation

The k-fold cross-validation (CV) is a general procedure for estimating the prediction error of any supervised classifier [55] and thus can be utilized, instead of the OOB estimate, for the purpose of automatization of the search for d_{sub} and the stopping criterion for the number of base learners L. In order to do that, we need to divide the training set into k equally populated folds at the very beginning of the training process. Instead of training each base learner on a bootstrap sample (roughly 63% unique samples of the training set) and evaluating it on the out-of-bootstrap sample points, it will be trained on $k-1$ folds and evaluated on the remaining fold. This will be repeated k times, leaving subsequently all the folds out and using them for error estimation.

We will call the OOB-based and the CV-based ensemble implementation the *OOB-ensemble* and the *CV-ensemble*, respectively. Both variants are quite similar, and differ in the following. After the first k base-learner trainings (the first base learner for each fold), the CV-ensemble returns exactly one vote for each training sample, as each of them was left out exactly once. On the other hand, the number of votes per training sample after the first k base-learner trainings of the OOB-ensemble follows the binomial distribution with the mean around $0.37k$, as roughly 37% of the training samples are left out in every round. So, as the ensemble training proceeds and after training the total of n base learners, the individual predictions of the CV error estimate are formed at every moment from n/k votes (for example $0.2n$ votes in case of five-fold cross-validation), while in the case of the OOB error estimate, they are formed from roughly $0.37n$ votes on average. Consequently, the OOB error estimate typically converges faster, for the price that the individual base learners are trained on a smaller number of unique samples, and the training points do not have identical number of votes, unlike in case of CV.

Another difference is that in the CV approach, the folds are formed at the beginning and remain the same for the whole training process, while in the OOB approach, a new bootstrap sample is drawn every time, yielding higher diversity.

The complexity of both approaches is similar.

3.5.1 Experimental comparison

We implemented the CV-ensemble, and experimentally compared its performance to the OOB-ensemble. There are two important points regarding the CV implementation we would like to make.

First, the folds need to be created in a way that the cover-stego pairs are preserved, i.e., the pairs of features coming from cover and the corresponding stego images should not be separated into two different folds. We pointed out this potential pitfall already in Section 3.1 – if the pairs were not preserved, the resulting CV error estimate might be heavily biased and the overall performance of the ensemble would be sub-optimal. We discuss this in more detail in Appendix C.3.

Second, as the training process of the CV-ensemble can be reinterpreted as training k parallel ensembles, the question is: should we use the same random subspaces for those k parallel ensembles? In other words, should we always use the same random subspace for every k-tuple of the subsequent base learners? Even though it would make sense to do it this way in order to give each of the training point votes from the same subspaces, it turns out that generating a new random subspace every time gives slightly better results. This is not surprising as it yields better diversity. Thus, we generate new random subspaces every time.

Experiments were conducted on 6,500 JPEG images from the CAMERA image database described in detail in Appendix B. The images were randomly divided into two halves for training and testing, respectively.

We steganalyzed three different JPEG domain steganographic algorithms: MBS [116], YASS [118], and nsF5[6] [51], covering a wide range of payloads (or YASS settings). These three algorithms

[6]The stego images were obtained using an nsF5 simulator available at http://dde.binghamton.edu/download/nsf5simulator/.

Steganalysis of MBS using CC-PEV features

payload	MED		MAD	
(bpac)	OOB	CV	OOB	CV
0.01	0.3849	0.3874	0.00260	0.00195
0.02	0.2810	0.2827	0.00340	0.00185
0.03	0.1966	0.1970	0.00175	0.00240
0.04	0.1271	0.1277	0.00220	0.00180
0.05	0.0815	0.0825	0.00105	0.00220

Steganalysis of YASS using CDF features

payload	MED		MAD	
(bpac)$^+$	OOB	CV	OOB	CV
0.187 (3)	0.0230	0.0229	0.00080	0.00160
0.138 (8)	0.0753	0.0743	0.00125	0.00075
0.159 (10)	0.0514	0.0500	0.00240	0.00255
0.114 (11)	0.0718	0.0718	0.00150	0.00085
0.077 (12)	0.1281	0.1288	0.00080	0.00185

$^+$The number in parentheses denotes YASS setting

Steganalysis of nsF5 using $\mathcal{CF}^*$ features

payload	MED		MAD	
(bpac)	OOB	CV	OOB	CV
0.05	0.3393	0.3393	0.00155	0.00245
0.10	0.1727	0.1734	0.00200	0.00155
0.15	0.0726	0.0714	0.00140	0.00115
0.20	0.0264	0.0285	0.00085	0.00050

Table 3.1: Steganalysis of MBS, YASS, and nsF5 using three different feature sets. Two different implementations of the ensemble classifier are compared – the OOB-ensemble and the CV-ensemble. We report the median (MED) of the classification error P_{E} and its median absolute deviation (MAD) over 10 different splits of the CAMERA database into a training and a testing set.

represent three different embedding paradigms: MBS is a model-preserving technique, YASS is a robust embedding that masks its impact by subsequent JPEG compression, and nsF5 minimizes the embedding impact. More details about the inner workings of these techniques can be found in Appendix A.

We used three different feature sets: CC-PEV [77] to detect MBS, CDF [82] to detect YASS (settings 3, 8, 10, 11, and 12 as reported in [82]), and the 7,850-dimensional $\mathcal{CF}^*$ set [81] to detect nsF5. These choices were made to cover a wide spectrum of feature types and dimensions.

We trained both types of ensemble classifiers (OOB and CV) for every payload or YASS setting separately, repeated all the experiments over 10 different splits of the CAMERA database into a training and testing set, and report the obtained median (MED) testing errors and the median absolute deviation (MAD) values in Table 3.1. We conclude that both implementations of the ensemble classifier yield similar results and thus either of them can be used.

3.6 Incorporating AdaBoost

The second direction of our efforts was to incorporate the ideas of AdaBoost [38] that trains base learners sequentially, continuously adjusts the weights of training samples based on their difficulty and forms the final decision as a weighted combination of individual votes. We now formalize the AdaBoost algorithm.

Given the training set $\mathcal{S}^{\text{trn}} = \{\mathbf{x}_m, \mathbf{y}_m | m \in \mathcal{I}^{\text{trn}}\}$, let $w^{(l)}_{\mathbf{x}_m}, w^{(l)}_{\mathbf{y}_m}$, $m \in \mathcal{I}^{\text{trn}}$ be the corresponding weights of the training cover samples ($\mathbf{x}_m$) and stego samples ($\mathbf{y}_m$) after the lth base learner B_l is trained. Alternatively, we may use the vector notation $\mathbf{w}^{(l)}_{\mathbf{x}}, \mathbf{w}^{(l)}_{\mathbf{y}} \in \mathbb{R}^{N^{\text{trn}}}$. We keep treating cover

and stego samples separately as it will be useful later in this section. The weights are initialized as

$$w_{\mathbf{x}_m}^{(0)} = w_{\mathbf{y}_m}^{(0)} = \frac{1}{2N^{\text{trn}}},\ \forall m \in \mathcal{I}^{\text{trn}}. \tag{3.6.1}$$

The lth base learner B_l, $l = 1, 2, \ldots$, is trained on $\mathcal{S}^{\text{trn}}$ with weights $\mathbf{w}_{\mathbf{x}}^{(l-1)}, \mathbf{w}_{\mathbf{y}}^{(l-1)}$. Its error may be expressed as

$$\epsilon_l = \frac{1}{2N^{\text{trn}}} \sum_{m \in \mathcal{I}^{\text{trn}}} \left(B_l(\mathbf{x}_m) + 1 - B_l(\mathbf{y}_m) \right), \tag{3.6.2}$$

where $B_l(\mathbf{x})$ is the cover (0) or stego (1) prediction of the base learner on $\mathbf{x}$. The weight of the lth *base learner* B_l is defined as

$$\alpha_l = \frac{1}{2} \ln \left(\frac{1 - \epsilon_l}{\epsilon_l} \right). \tag{3.6.3}$$

Once the lth base learner is trained, the training sample weights are updated as

$$w_{\mathbf{x}_m}^{(l)} = \frac{1}{Z_l} \cdot e^{-\alpha_l (-1)^{B_l(\mathbf{x}_m)}},\ m \in \mathcal{I}^{\text{trn}}, \tag{3.6.4}$$

$$w_{\mathbf{y}_m}^{(l)} = \frac{1}{Z_l} \cdot e^{+\alpha_l (-1)^{B_l(\mathbf{y}_m)}},\ m \in \mathcal{I}^{\text{trn}}, \tag{3.6.5}$$

where Z_l is the normalization factor ensuring that $\sum_{m \in \mathcal{I}^{\text{trn}}} w_{\mathbf{x}_m}^{(l)} + w_{\mathbf{y}_m}^{(l)} = 1$.

After L base learners are trained, the final ensemble prediction on a given (testing) sample $\mathbf{y}$ is formed as

$$B(\mathbf{y}) = \begin{cases} 0 & \text{if } \frac{1}{Z_\alpha} \sum_{l=1}^{L} \alpha_l B_l(\mathbf{y}) < 0.5, \\ 1 & \text{if } \frac{1}{Z_\alpha} \sum_{l=1}^{L} \alpha_l B_l(\mathbf{y}) > 0.5, \\ \text{random} & \text{otherwise}, \end{cases} \tag{3.6.6}$$

where $Z_\alpha = \sum_{l=1}^{L} \alpha_l$.

3.6.1 Application to steganalysis

The AdaBoost algorithm as described above can be applied to the ensemble framework proposed in Section 3.1 in several ways.

The first option is to boost individual FLDs using AdaBoost, and use these boosted FLDs, each of them trained in a different feature space and on a different bootstrap sample of the training set, as the base learners for the final ensemble framework. This idea was used for example in [136], where the authors applied it to image classification. We would need to replace the standard Fisher Linear Discriminant with its weighted counterpart because the standard FLD does not accept weighted training samples at the input. This can be done, for example, as described in [88] using generalized definitions of a mean and a covariance matrix that incorporate sample weights. However, we would lose the ability to accurately estimate the testing error of the ensemble because the continuously monitored predictions of individual training samples would now be formed as different *weighted* combinations of different *number* of votes and thus very few of these combined predictions would be formed in the same way as the final testing predictions would be. This was not an issue when we used the majority voting strategy of equally important predictions as there we had a guarantee that none of the predictions used for error estimation missed an *important* vote or was formed only by votes that were not important. Furthermore, even though the complexity of the generalized FLD is similar to the one of the standard FLD, the complexity of the whole system would multiplicatively grow by the number of iterations needed by AdaBoost for boosting every single FLD, which is not desirable.

The second option is to use the whole ensemble as a single base learner for AdaBoost, which is the other way of assembling AdaBoost and bagging. However, the complexity of this approach would grow in the same manner as with the previous idea, and even here we would lose the convenience of estimating the testing error simply as OOB estimates, and some sort of additional cross-validation would need to be used.

The last option is to incorporate the ideas of AdaBoost into the framework *directly*, i.e., to keep adjusting the training sample weights as the training progresses and form the final testing predictions according to the rule (3.6.6). However, unlike in the original AdaBoost, there are two non-trivial problems that need to be resolved as every base learner is trained:

1. in a different feature space (in the random subspace of the original space $\mathcal{F}$).
2. on different training samples (bootstrap samples from the original training set).

The first point, a different feature space every time a base learner is trained, can be seen as a property of the base learner itself – each learner is able to utilize only a portion of the feature space. This may be a problem in situations where a small number of features is responsible for majority of the classification accuracy because the "importance" of a given training sample $\mathbf{x}_m$ (or $\mathbf{y}_m$), described by a single weight $w_{\mathbf{x}_m}$ (or $w_{\mathbf{y}_m}$), may depend on the feature space. Fortunately, this does not happen in modern steganalysis where the power of a feature space is typically spread across all the features (unless the steganography is fatally flawed and/or the cover source is somehow singular).

The second point is more challenging – the lth base learner is trained on $\mathcal{S}_l^{\text{trn}} = \{\mathbf{x}_m, \mathbf{y}_m | m \in \mathcal{N}_l^{\text{b}}\}$, where $\mathcal{N}_l^{\text{b}}$ is the lth bootstrap sample of the set of indices $\mathcal{I}^{\text{trn}}$, and therefore roughly 37% of the training cover-stego pairs are omitted in its training. This poses the following questions. Using formulas (3.6.4) and (3.6.5), should we update the weights of all the training samples from $\mathcal{S}^{\text{trn}}$ or only those that belong to $\mathcal{S}_l^{\text{trn}}$? When calculating the error ϵ_l in order to obtain the base-learner weight α_l, the same question arises – should we use all the training samples or only those that belong to $\mathcal{S}_l^{\text{trn}}$? Furthermore, the situation gets complicated due to the fact that every time a *different* set of 37% samples is omitted. We could either use all the training samples for updating, knowing that some weights may be adjusted incorrectly, as some points would be classified differently if they were part of the training, or the weights would be updated unevenly, only to the samples from $\mathcal{S}_l^{\text{trn}}$ in the lth iteration. This may influence the convergence properties of the ensemble and, more importantly, it would have a negative impact on the accuracy of OOB estimates, a crucial element of the system needed for the determination of parameters d_{sub} and L.

An appealing (and simple) way of resolving this problem is to use the cross-validation variant of the ensemble described in Section 3.5. Since the folds in k-fold cross-validation are formed at the very beginning and remain the same for the rest of the training process, every k-th base-learner is trained *exactly* on the same training samples, i.e., on all folds but the kth one. Therefore, viewing the entire process as training k parallel sub-machines, we could apply AdaBoost to each of them individually by keeping track of k different sets of weights. This way, each of the boosted sub-ensembles produces $1/k$ of equally important predictions for error estimating purposes, the CV error estimate corresponds to the way testing predictions will be made, and the algorithms for automatic determination of parameters d_{sub} and L can be used as before with the caveat that we use the CV error estimates instead of the OOB estimates.

The implementation of the system is straightforward: we need k sub-classifiers trained in parallel, each of them implemented as weighted FLDs boosted by AdaBoost as described at the beginning of this section. After every step of the training process, the predictions of the folds left out are updated using the weighted rules (3.6.6) and combined from all k folds together to form the updated value of the CV error estimate. The training stops once this error estimate converges, for which we may use the same stopping criterion as in the original ensemble algorithm – formula (3.2.2). We also apply the same search algorithm for finding the optimal value of d_{sub}.

Before we proceed to the experimental evaluation of the described system, we make the following comment. As already mentioned earlier, binary classifiers used for steganalysis should be trained on

Steganalysis of MBS using CC-PEV features

payload	MED		MAD	
(bpac)	OOB	Ada	OOB	Ada
0.01	0.3849	0.3871	0.00260	0.00140
0.02	0.2810	0.2833	0.00340	0.00220
0.03	0.1966	0.1977	0.00175	0.00235
0.04	0.1271	0.1301	0.00220	0.00200
0.05	0.0815	0.0870	0.00105	0.00215

Steganalysis of YASS using CDF features

payload	MED		MAD	
(bpac)$^+$	OOB	Ada	OOB	Ada
0.187 (3)	0.0230	0.0250	0.00080	0.00170
0.138 (8)	0.0753	0.0762	0.00125	0.00245
0.159 (10)	0.0514	0.0515	0.00240	0.00215
0.114 (11)	0.0718	0.0756	0.00150	0.00170
0.077 (12)	0.1281	0.1359	0.00080	0.00160

$^+$The number in parentheses denotes YASS setting

Steganalysis of nsF5 using $\mathcal{CF}^*$ features

payload	MED		MAD	
(bpac)	OOB	Ada	OOB	Ada
0.05	0.3393	0.3402	0.00155	0.00215
0.10	0.1727	0.1815	0.00200	0.00330
0.15	0.0726	0.0853	0.00140	0.00220
0.20	0.0264	0.0377	0.00085	0.00105

Table 3.2: Steganalysis of MBS, YASS and nsF5 using three different feature sets. Two different implementations of the ensemble classifier are compared – the OOB-ensemble and the CV-ensemble boosted with AdaBoost (column 'Ada'). We report the median (MED) of the classification error P_{E} and its median absolute deviation (MAD) over 10 different splits of the CAMERA database into a training and a testing set.

pairs of cover-stego feature vectors. Keeping this in mind, not only do we need to create the folds for cross-validation in a way to preserve these pairs of feature vectors, but we also have to modify the weights-updating rules of the classical AdaBoost. In the original formulas (3.6.4) and (3.6.5), the weight of every training sample is increased if it is misclassified by the current base learner, and it is decreased if the sample is classified correctly. We modify the update procedure as follows. If *both* features from every cover-stego pair are classified correctly, their weight is decreased. If at least one of them is misclassified, the weight of *both* of them is increased. This is a logical modification of the AdaBoost for steganalysis that guarantees that the focus of every subsequent base learner will be on more and more difficult training samples, while at the same time preserving the intrinsic cover-stego pairing.

3.6.2 Experimental comparison

We implemented the AdaBoost-based version of the ensemble classifier and subjected it to a steganalysis test, comparing its performance to the standard OOB-ensemble described in Section 3.1. Experiments were conducted in a similar manner as in Section 3.5. In particular, we used the same image database and steganalyzed the same three algorithms – MBS, YASS, and nsF5 using the feature sets CC-PEV, CDF, and $\mathcal{CF}^*$, respectively. We trained the ensemble classifier for every payload or YASS setting separately. The comparison is shown in Table 3.2 in terms of the median error P_{E} over 10 different splits of the CAMERA database into a training and testing set.

We conclude that boosting the ensemble through AdaBoost does not bring any performance gain. It even seems to deliver slightly worse results than the original ensemble implementation, which is more apparent for larger payloads when the classes are more distinguishable. Therefore, we recommend

bpac	Classifier	P_E MED	P_E MAD	Training time
0.05	G-SVM	0.3772	0.00246	7 hr 29 min
	L-SVM	0.3802	0.00269	23 min
	Ensemble	0.3695	0.00145	2 min
0.10	G-SVM	0.2326	0.00231	6 hr 52 min
	L-SVM	0.2421	0.00246	27 min
	Ensemble	0.2226	0.00265	3 min
0.15	G-SVM	0.1247	0.00385	5 hr 39 min
	L-SVM	0.1342	0.00185	26 min
	Ensemble	0.1160	0.00170	3 min
0.20	G-SVM	0.0615	0.00200	4 hr 51 min
	L-SVM	0.0638	0.00123	27 min
	Ensemble	0.0547	0.00150	3 min

Table 3.3: Steganalysis of nsF5 using CC-PEV features. The running times and P_E values are medians (MED) over ten independent splits of the CAMERA database into training and testing sets. We also report median absolute deviation (MAD) values for P_E.

to implement the ensemble as a random forest built from equally weighted FLDs trained on different random subspaces as described in Section 3.1.

The suboptimality of the boosted ensemble is probably caused by an inappropriate combination of random subspaces and weighted voting. The original OOB-ensemble is a random forest, and each of its base learners has a random parameter that is drawn independently and from the same distribution for all base learners (random subspace generation). Therefore, it makes sense to treat each base learner *equally* in the final voting. However, once we incorporate AdaBoost, the first few base learners produce the most important votes for the final decision because they are trained on the training samples with similar weights. On the other hand, later base learners are trained on an increasingly more difficult training set, producing classifiers with lower accuracy and thus lower weight of their vote. The cover/stego class distinguishing ability of random subspaces formed later in the training process has therefore lower influence on the final decision than random subspaces formed at the beginning of the training, in spite of their equal forming strategy.

3.7 Comparison to SVMs

The ensemble classifier is proposed as an alternative tool to SVMs for feature development in steganalysis and for building steganalyzers. In this last section of Chapter 3, we compare it with both Gaussian and linear SVMs in terms of training complexity and performance.

As the feature-space dimensionality and the number of training samples increase, the complexity and memory requirements of SVMs increase quite rapidly. The training complexity scales at least quadratically with N^{trn} and grows as the cost parameter[7] C increases or as the classes become less distinguishable. Performing a proper k-fold cross-validation over the pre-defined grid of the penalty parameter $C \in \mathcal{G}_C$ requires repeating the training $k \cdot |\mathcal{G}_C|$ times. Furthermore, in case of the non-linear G-SVM, the grid of hyper-parameters is two-dimensional as we also need to search

[7]The parameter C is a cost for misclassification in the objective function of SVM training.

N^{trn}	G-SVM	L-SVM	Ensemble
1,000	33 min	5 min	$<$ 1 min
2,000	2 hr 27 min	10 min	1 min
3,000	5 hr 24 min	14 min	1.5 min
4,000	9 hr 31 min	20 min	2 min
5,000	13 hr 47 min	27 min	2 min
10,000	$\times$	54 min	4 min
15,000	$\times$	1 hr 23 min	5 min
20,000	$\times$	1 hr 52 min	6 min
25,000	$\times$	2 hr 21 min	8 min

Table 3.4: Dependence of the training time on N^{trn}. Target algorithm: nsF5 with 0.10 bpac.

for the optimal value of the kernel width $\gamma \in \mathcal{G}_\gamma$, which makes the training even more expensive. Additionally, a G-SVM needs the kernel matrix of size $(N^{\text{trn}})^2$ to be stored in the memory, which becomes prohibitive even for moderately large training sets and requires a more advanced caching solution.

In contrast, training the ensemble classifier requires much more moderate computer resources. To compute the scatter matrix $\mathbf{S}_{\text{W}}$ (3.1.4) for one base learner, $O(N^{\text{trn}} d_{\text{sub}}^2)$ operations are needed while the matrix inversion in (3.1.2) can be carried out in $O(d_{\text{sub}}^3)$ operations. Thus, the total training complexity for a fixed value of d_{sub} is $O(LN^{\text{trn}} d_{\text{sub}}^2) + O(Ld_{\text{sub}}^3)$, which is linear w.r.t. N^{trn} and does not directly depend on d. As the search for the optimal value of d_{sub} (described in Section 3.2) is performed essentially by increasing the value of d_{sub} until the optimal value is found, the complexity is dominated by the largest value of the random subspace dimensionality tried during the search, which is always close to the optimal d_{sub}. We also note that, unlike SVM, FLD is scale-invariant and the features do not need to be normalized.

With respect to the memory requirements, each base learner needs access to only $2N^{\text{trn}} d_{\text{sub}}$ features at a time. Therefore, the classifier could be implemented so that one never needs to load all d features into the memory during training, which is a favorable property especially for very large d. The ensemble classifier is represented using the set of L generalized eigenvectors (3.1.2) and the corresponding thresholds, which requires the total storage space of $O(Ld_{\text{sub}})$.

In the first experiment, we attack the steganographic algorithm nsF5 using the 548-dimensional CC-PEV features. We use this fairly low-dimensional feature set[8] so that we can easily train the following three classifiers without running into complexity issues:

- Gaussian SVM implemented using the publicly available package LIBSVM [21]. The training includes a five-fold cross-validation search for the optimal hyper-parameters – the cost parameter C and the kernel width γ. It was carried out on the multiplicative grid $\mathcal{G}_C \times \mathcal{G}_\gamma$, $\mathcal{G}_C = \{10^a\},\ a \in \{0, \dots, 4\}$, $\mathcal{G}_\gamma = \left\{\frac{1}{d} \cdot 2^b\right\},\ b \in \{-4, \dots, 3\}$.
- Linear SVM implemented using the package LIBLINEAR [30]. The five-fold cross-validation is used to search over the grid of the cost parameter $C \in \{10^a\},\ a \in \{-4, \dots, 3\}$.
- Ensemble classifier implemented in Matlab as described in Section 3.1, including the search for the optimal value of d_{sub} and the automatic determination of L. Our implementation is available for download at `http://dde.binghamton.edu/download/ensemble/`.

[8] Note the shift in the notion of what constitutes a *low*-dimensional feature set.

	L-SVM		Ensemble	
	P_{E}	time	P_{E}	time
nsF5 0.05	0.3518	9 hr 27 min	0.3377	31 min
nsF5 0.10	0.1851	8 hr 04 min	0.1737	37 min
nsF5 0.15	0.0809	6 hr 36 min	0.0720	24 min
nsF5 0.20	0.0292	5 hr 37 min	0.0273	15 min
YASS 3	0.0222	5 hr 47 min	0.0146	48 min
YASS 8	0.0377	6 hr 12 min	0.0271	45 min
YASS 10	0.0241	5 hr 35 min	0.0164	59 min
YASS 11	0.0616	5 hr 46 min	0.0437	58 min
YASS 12	0.0711	6 hr 12 min	0.0532	1 min
MB 0.01	0.3806	13 hr 39 min	0.3710	43 min
MB 0.02	0.2654	14 hr 58 min	0.2560	1 hr 9 min
MB 0.03	0.1820	13 hr 52 min	0.1684	57 min
MB 0.04	0.1094	11 hr 23 min	0.1087	59 min
MB 0.05	0.0715	10 hr 24 min	0.0684	49 min

Table 3.5: Steganalysis using $\mathcal{CF}^*$ features. The L-SVM classifier is compared with the proposed ensemble classifier.

Splitting the CAMERA database into two halves – one for training and the other half for testing, a set of stego images was created for each relative payload $\alpha \in \{0.05, 0.1, 0.15, 0.2\}$ bpac. Table 3.3 shows the detection accuracy and the training time of all three classifiers. The time was measured on a computer with the AMD Opteron 275 processor running at 2.2 GHz.

The performance of all three classifiers is very similar, suggesting that the optimal decision boundary between cover and stego images in the CC-PEV feature space is linear or close to linear. Note that while the testing errors are comparable, the time required for training differs substantially. While the G-SVM took several hours to train, the training of L-SVMs was accomplished in 14–23 minutes. The ensemble classifier is clearly the fastest, taking approximately 2 minutes to train across all payloads.

With the increasing complexity and diversity of cover models, future steganalysis must inevitably start using larger training sets. Our second experiment demonstrates that the computational complexity of the ensemble classifier scales much more favorably w.r.t. the training set size N^{trn}.[9] To this end, we fixed the payload to 0.10 bpac and extended the CAMERA database by $10,000$ images from the BOWS2 competition and $9,074$ BOSSbase images (see Appendix B for more details). Both databases were JPEG compressed using the quality factor 75. The resulting collection of images allows us to increase the training size to $N^{\text{trn}} = 25,000$.

Table 3.4 shows the training times for different N^{trn}. The values for the L-SVM and the ensemble classifier are averages over five independent random selections of training images. As the purpose of this experiment is rather illustrative and the G-SVM classifier is apparently computationally infeasible even for relatively low values of N^{trn}, we report its training times measured only for a single randomly selected training set and drop it from experiments with $N^{\text{trn}} > 5,000$ entirely. It is apparent that although the L-SVM training is computationally feasible even for the largest values of N^{trn}, the ensemble is still substantially faster and the difference quickly grows with the training set size.

In our last experiment, we compare the performance of the L-SVM with the ensemble classifier when a high-dimensional feature space is used for steganalysis. The G-SVM was not included in this test

[9] The actual number of training samples is $2N^{\text{trn}}$ as we take cover-stego pairs.

due to its high complexity. We consider the $7,850$-dimensional feature vector $\mathcal{CF}^*$ and use it to attack nsF5, YASS, and MBS. This experiment reveals how well both classifiers handle the scenario when the feature space dimensionality is larger than the number of training samples ($2 \times 3,250$). The comparison is reported in Table 3.5. The ensemble classifier is substantially faster and delivers a higher accuracy for all three algorithms (the decision boundary seems to be farther from linear than in the case of CC-PEV features).

3.8 Summary

The current trend in steganalysis is to train classifiers with increasingly more complex cover models and large data sets. However, the machine learning tool of choice, the support vector machine, does not scale favorably w.r.t. feature space dimensionality and the training set size which poses serious constraints on the steganalysts. In particular, the steganalyst is forced to keep the number of training images rather small and feature spaces are *designed* to be low-dimensional from the very beginning. Consequently, the feature space design becomes a tedious task based on clever tricks and heuristic dimensionality-reduction techniques, making the proposed steganalysis feature sets difficult to interpret.

Machine learning should *not* be an obstacle for the construction of feature spaces and therefore we propose an alternative – ensemble classifiers built by fusing decisions of weak and unstable base learners implemented as the Fisher Linear Discriminant. Performance-wise, ensemble classifiers offer accuracy comparable to the much more complex SVMs at a fraction of the computational cost. Furthermore, they scale more favorably w.r.t. both the dimensionality of the feature space and the number of training images.

The benefit of the ensemble is twofold. First, it is useful for fast feature development when attacking a new scheme. For example, during the recent steganalysis competition BOSS [9], our team was able to test hundreds of ideas, quickly eliminating most of them while keeping and further exploring only the most promising ones. Second, as the feature space design is no longer restricted by dimensionality, cover models can be constructed more systematically and transparently, resulting in rich feature spaces capturing many different dependencies among cover coefficients. This will be demonstrated in Chapters 5 and 6 of this dissertation.

Chapter 4

Understanding feature spaces

The accuracy of feature-based steganalyzers is inevitably linked to the quality of constructed feature spaces (image models). Understanding the principles for design of feature spaces and their intrinsic properties is therefore of a great importance for steganalysts' success. This chapter covers various aspects of feature space construction, starting by introducing general guidelines in Section 4.1 and demonstrating them on specific examples of feature sets that have been proposed in literature.

In Section 4.2, we cover the important concept of calibration that was introduced in 2002 [44] and later incorporated into many feature-based steganalysis systems [107, 92, 131, 59]. While calibration improves the detection performance, we challenge the generally accepted thesis that its purpose is to estimate the cover image features from the stego image and shed more light on how, why, and when calibration works. Furthermore, we propose a modified calibration procedure, the so-called Cartesian calibration, and show its superiority over the original calibration.

The never-ending battle between steganography and steganalysis could be seen as a competition for a better model – Alice tries to embed undetectably within her model while Eve tries to identify those relationships among cover coefficients that have been neglected by Alice and incorporates them into a new model. Developments in steganography and steganalysis are thus strongly interrelated and the modern feature space design reflects the successes and failures of both. In Section 4.3, we discuss a common flaw of many proposed steganographic techniques – overtraining to an imperfect cover model. We will show that even though recent advances in the direction of minimal-distortion steganography allow Alice to construct practical data hiding schemes that approximately preserve a given feature vector [36, 32], the design of secure stegosystems still remains rather difficult. Furthermore, deficiencies of existing stego-schemes show us what needs to be avoided when designing new feature spaces.

We summarize the gained insight in Section 4.4.

4.1 Fundamental design principles

Based on our experience and intuition gained over the years of research in the field of steganalysis, in this section we provide several general guidelines for construction of feature spaces. Before we do so, however, let us make a few comments on steganalysis from a broader perspective. According to the definition of the steganographic channel in Chapter 2.1, Eve attempts to detect a specific steganographic algorithm employed by Alice and Bob. This scenario is commonly known as *targeted* steganalysis, as the attack is targeted to the given stegosystem. A small number of features may be sufficient for a satisfactory detection performance. Such features, however, would unlikely perform well on a different steganographic scheme. If Eve's goal is to design a feature set that would be capable of detecting a wide range of steganographic algorithms, we speak of *blind* steganalysis.

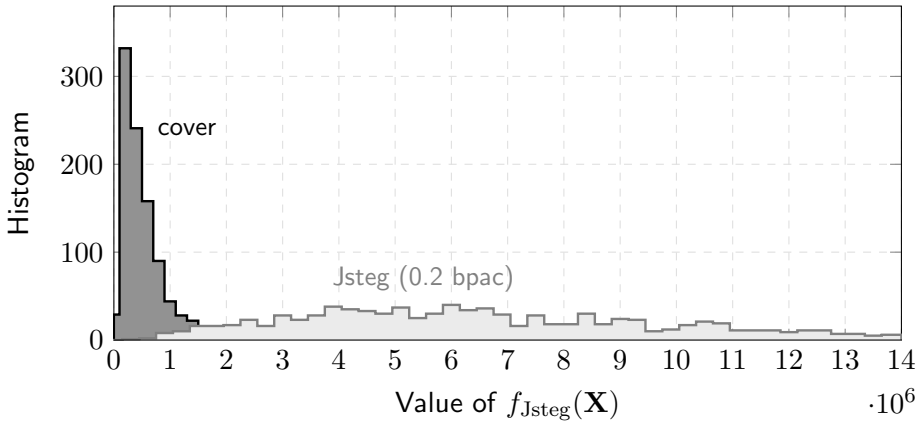


Figure 4.1.1: Histogram of $f_{\text{Jsteg}}(\mathbf{X})$ over 1000 cover images (dark) and 1000 Jsteg stego images (light). Images were randomly selected from the CAMERA database.

Another term that often appears in the literature is *universal* steganalysis – while not defined formally, here the goal is to construct a detector that would be able to detect all possible stegosystems, including those previously unseen. We note that the terms 'universal' and 'blind' are often used interchangeably.

In this dissertation, we would like to de-emphasize the differences among targeted, blind, and universal types of steganalysis as they overlap substantially in terms of the feature-space design. To be more specific, feature spaces in blind steganalysis are often inspired by targeted attacks as one can easily include statistics derived from targeted attacks as additional features, making the feature space richer. At the same time, while it is true that targeted steganalysis could do a good job with just a small number of features, it is also known that extending the targeted feature space by additional statistical descriptors of image-coefficients' dependencies usually further increases detection accuracy. Furthermore, we believe that the problem of designing a universal steganalyzer should be more about how to train the classifier rather than how to design its feature space. Obviously, the more dependencies among coefficients of natural images are captured, the better performance we could expect on unseen steganographic techniques, as we do not know in advance which of the dependencies would be disturbed by embedding. What is not so straightforward, however, is how we should *train* the steganalyzer in that feature space. Should we train it on examples of a large collection of different steganographic schemes (discriminative approach) or train a one-class machine only on examples of cover images (generative approach)? As the focus of this work is on the feature-space design, we leave this relevant question unanswered and refer the reader to a few works exploring these directions [110, 111, 106].

4.1.1 Known cover or stego properties

When attacking an algorithm, one of the first steps is finding a quantity whose value is known for cover images and that predictably changes with embedding. For example, the histogram of DCT coefficients of a natural JPEG image is known to be symmetric around zero and there are steganographic techniques, for example Jsteg [129], that disturb this symmetry. A quantity that captures the histogram symmetry would be therefore a good candidate as a feature for steganalysis of Jsteg. For example, the square difference between the positive and the negative part of the histogram could serve as a good feature:

$$f_{\text{Jsteg}}(\mathbf{X}) = \sum_{i>0} \left(h_i(\mathbf{X}) - h_{-i}(\mathbf{X})\right)^2, \tag{4.1.1}$$

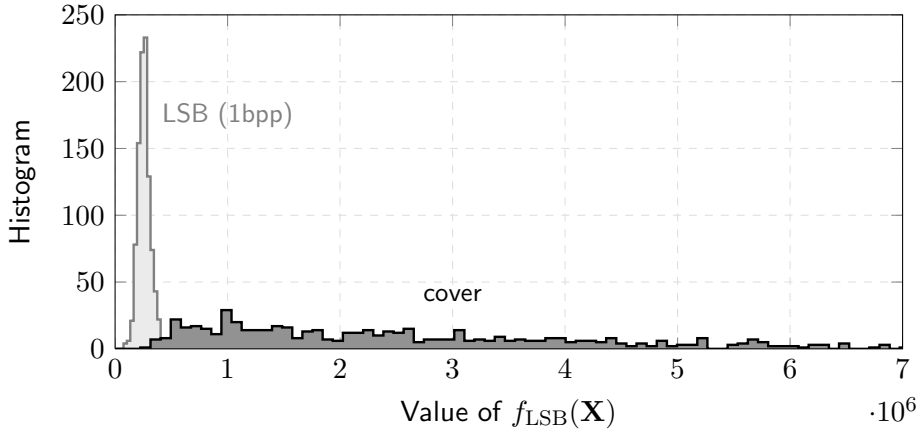


Figure 4.1.2: Histogram of $f_{\mathrm{LSB}}(\mathbf{X})$ over 1000 cover images (dark) and 1000 LSB stego images (light). Images were randomly selected from BOSSbase v0.92 database.

where $h_i(\mathbf{X})$ is the number of DCT coefficients of value i in the JPEG image $\mathbf{X}$ (ith histogram bin). For illustration, in Figure 4.1.1 we compare the histogram of $f_{\mathrm{Jsteg}}(\mathbf{X})$ values over 1000 cover images versus 1000 Jsteg stego images. The relative payload was fixed to 0.2 bits per nonzero AC DCT coefficient (bpac) and the images were randomly selected from the CAMERA database (see Appendix B). We can nicely see the distinguishing power of the scalar feature $f_{\mathrm{Jsteg}}(\mathbf{X})$ – in fact, if we were to steganalyze Jsteg simply by thresholding the value of $f_{\mathrm{Jsteg}}(\mathbf{X})$, we would be able to push the classification error P_{E} below 5%.[1]

Alternatively, one may identify a feature whose value is known for *stego* images and is different for covers. Let us consider LSB embedding in the spatial domain (Appendix A.11) which flips the least significant bits of visited pixels. In a fully embedded image, approximately half of the pixels is modified – even pixel values $2k$ are flipped to the value $2k+1$ and vice versa, values $2k+1$ are flipped to $2k$. As a result, the histogram values of all the LSB pairs $[2k, 2k+1]$, $k = 0, \ldots, 127$ are approximately equalized. We can define the following quantity as a "measure of equality of LSB pairs":

$$f_{\mathrm{LSB}}(\mathbf{X}) = \sum_{k=0}^{127} \left(h_{2k}(\mathbf{X}) - h_{2k+1}(\mathbf{X})\right)^2, \tag{4.1.2}$$

where $h_i(\mathbf{X})$ is the number of pixels of value i in the spatial domain image $\mathbf{X}$. In Figure 4.1.2, we can see the histogram of $f_{\mathrm{LSB}}(\mathbf{X})$ values over 1000 cover images and 1000 fully embedded LSB stego images, randomly selected from the BOSSbase v0.92 database (see Appendix B). The distinguishing power is apparent – this time, thresholding $f_{\mathrm{LSB}}(\mathbf{X})$ would give us the classification error P_{E} below 2%.

4.1.2 Adopting a simplified model

Even though digital images are *not* i.i.d. sequences, nothing prevents us from adopting the histogram of pixel values as a simplified model of natural images and use it for steganalysis. We have already seen that first order statistics may provide performance better than random guessing. If we start considering higher order dependencies, one usually further improves the detection performance. In

[1] The threshold can be learned from a set of training images.

Feature	Notation	Dimensionality
Global histogram	$\mathbf{h} = (h_i)$	11
5 AC histograms	$\mathbf{a}^{(k,l)} = (a_i^{(k,l)})$	$5 \times 11 = 55$
11 dual histograms	$\mathbf{d}^{(k,l)} = (d_i^{(k,l)})$	$11 \times 9 = 99$
Variation	v	1
Two types of blockiness	$\mathbf{b} = (b_i)$	2
Co-occurrence matrix	$\mathbf{C} = (C_{ij})$	$5 \times 5 = 25$
Markov features	$\mathbf{M} = (M_{ij})$	$9 \times 9 = 81$
Total dimensionality:		274

Table 4.1: List of all features from the PEV feature set.

[123], for example, the authors assume that differences between neighboring DCT coefficients form a Markov process and construct features as transition probability matrices. In the spatial domain, a similar approach resulted in the SPAM features [104] that were powerful against ±1 embedding.

Adopting a simplified model of cover images can be turned into a powerful feature building strategy if we start merging different simple models together. For instance, in [22] the authors extended their previous inter-block model by considering also intra-block dependencies. Another example of this merging strategy is the popular 274-dimensional PEV feature space [107] formed as a combination of Markov features [123] and 193 various DCT based statistics which, by themselves, are formed as a merger of different smaller submodels – global histogram, local AC histograms, inter-block co-occurrences, etc. The complete list of submodels contained in the PEV feature space is shown in Table 4.1, while a more detailed description of the feature set can be found in the original publication [107]. This set has been used as an oracle for design of steganographic schemes [89, 126, 118], for performance comparisons [51, 75, 2, 115, 53, 26], and benchmarking [108].

4.1.3 Modeling noise

As the amplitude of steganographic modifications is usually small, embedding could be seen as adding a certain type of noise to the image. Therefore, rather than modeling the cover coefficients directly, features are often constructed to capture dependencies among their *noise residuals*. This way, the features' sensitivity to embedding changes is improved, while the undesirable sensitivity to the image content is suppressed. Examples of this strategy are WAM features [52] or the features proposed in [96], both calculated as higher-order statistics of the noise residual in the wavelet domain.

In a sense, all features modeling *differences* between neighboring coefficients, for example the previously mentioned Markov features [22] and the SPAM features [104], can be classified as noise-modeling approaches, as the coefficients themselves are simple predictors of their neighbors and thus the difference of two neighboring coefficients is the simplest possible residual.

In the recent publication [50], the authors turned the noise-modeling approach into a general and complex framework. They formed features as co-occurrences of image noise residuals obtained from higher-order local models of images. The resulting feature set, called HOLMES (Higher-Order Local Model Estimators of Steganographic changes) was used to attack the spatial domain steganographic algorithm HUGO [105] and was shown to perform well also on ±1 embedding.

Other examples of the philosophy of modeling noise residuals rather than image content are [31, 6, 5].

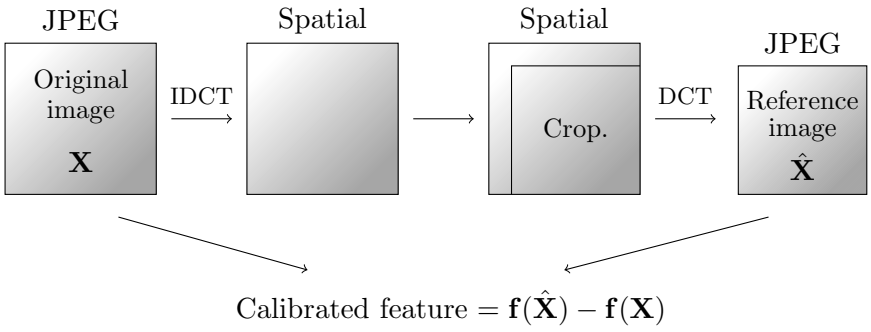


Figure 4.1.3: The process of calibration as incorporated in the PEV feature set.

4.1.4 Providing references

Steganalysis can often be improved by providing the classifier with the so-called *references* – additional features whose purpose is to supplement other features, already present in the feature set, with certain reference values. Example: Let us consider the Jsteg steganography, the DCT-histogram-symmetry violating technique mentioned earlier in this section. Jsteg does not embed into DCT coefficients whose value is equal to one. As an embedding invariant, the number of ones in a JPEG image, i.e., the histogram bin $h_1(\mathbf{X})$, may thus seem to be a useless statistic for distinguishing between cover images and Jsteg stego images. We could not be more wrong – the number of ones in a JPEG image is a very useful reference value to the histogram bin $h_{-1}(\mathbf{X})$. In a cover image, these two values are approximately equal due to histogram symmetry, however, during Jsteg embedding, $h_{-1}(\mathbf{X})$ rapidly decreases as Jsteg changes half of the visited coefficients -1 to the value -2. As a result, the histogram pair $[h_{-1}(\mathbf{X}), h_1(\mathbf{X})]$ has a significantly more informative value than just the single feature $h_{-1}(\mathbf{X})$, even though $h_1(\mathbf{X})$ does not change with embedding.[2]

Another reference-providing strategy is the concept of calibration, originally introduced in 2002 as a part of the attack on the F5 algorithm [44]. Calibration is an important element of modern feature-space design and therefore we decided to devote the whole next section solely to the study of calibration.

4.2 The concept of Cartesian calibration

The concept of calibration was used for the first time in [44], where the authors introduced it as a method to estimate the cover image histogram from the stego image when attacking the steganographic algorithm F5 [132]. The same mechanism was later used in [45] to successfully attack OutGuess [113]. Since then, calibration has been incorporated into many feature-based steganalyzers and has been shown to generally improve feature-based steganalysis [39, 107, 59, 92].

We challenge the thesis that the purpose of calibration is to estimate the cover image features from the stego image, and subjected calibration to a detailed analysis. As a result, we discovered that for small payloads, calibration, indeed, does not provide an estimate of the cover image features. In fact, the stego image features are often a much better approximation of the original cover features. This is quite strikingly apparent in WS steganalysis [41] where the predictor values are on average much farther from the cover than stego. Furthermore, we found situations when calibration *hurts* the detection performance.

[2] Note that in the formula (4.1.1), the histogram bin $h_1(\mathbf{X})$ is combined with $h_{-1}(\mathbf{X})$ to form the square difference. Here, we pass these two values to the machine-learning engine as two separate features, and let the classifier itself decide whether their square difference is the optimal way of combining them together.

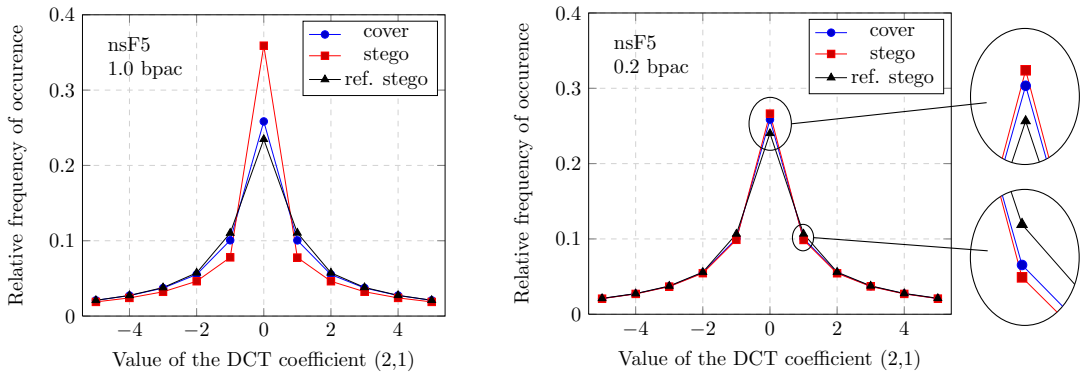


Figure 4.2.1: The effect of nsF5 embedding on the histogram of the DCT mode (2,1) for payloads 1.0 bpac (left) and 0.2 bpac (right). The graph was obtained as an average over all 6500 images in the CAMERA database.

We identified several different mechanisms that may be responsible for the beneficial effect of calibration, and published our findings in [77], where we also proposed a modified calibration procedure, the so-called Cartesian calibration, that has been shown to overcome the problems of the original way of calibrating features. As understanding calibration is vital for feature-space design, in this Section we now go through the most important parts of our publication [77].

4.2.1 Motivation

The idea behind calibration is to produce an image that would be perceptually similar to the cover image, the so-called *reference image*, and use its coefficient statistics to estimate the same statistics of the cover image. A *calibrated feature* is then defined as a difference between a feature extracted from the original image and the same feature extracted from the reference image.

In the spatial domain, a reference image can be created, for example, by downsampling [63, 64]. Figure 4.1.3 illustrates the process of calibration in the JPEG domain. It starts with a JPEG image $\mathbf{X} \in \mathcal{C}$ under investigation, decompresses it into the spatial domain using inverse DCT, crops by four pixels in both directions, and recompresses the cropped image using the quantization matrix of $\mathbf{X}$. As a result, a reference JPEG image, $\hat{\mathbf{X}} \in \mathcal{C}$, is obtained. The spatial domain cropping by 4 pixels was incorporated into the PEV feature set listed in Table 4.1 and is probably the most common form of calibration for JPEG images. As suggested in [39], however, very similar results are indeed obtained by applying a slight amount of rotation or resizing since such operations also desynchronize the original 8×8 grid, erasing thus the impact of embedding in the DCT domain. Even though the image $\hat{\mathbf{X}}$ can have slightly different dimensions from the original $\mathbf{X}$, this does not affect further feature extraction because the features are usually normalized. Once the reference image is created, the calibrated version of the feature vector $\mathbf{f}(\mathbf{X})$, given by a feature mapping $\mathbf{f} : \mathcal{C} \to \mathcal{F}$, is the difference $\mathbf{f}(\hat{\mathbf{X}}) - \mathbf{f}(\mathbf{X})$.

In [44], the authors include the following heuristic explanation why calibration works:

> "Cropping the image produces an image that is perceptually similar to the original and therefore its DCT coefficients should have approximately the same statistical properties as the DCT coefficients of the cover image. Furthermore, the spatial shift by four pixels ensures that the 8×8 grid of recompression *does not see* the previous JPEG compression and thus the obtained DCT coefficients are not influenced by previous quantization (and possible embedding) in the DCT domain. Therefore, the statistics of the reference image (its feature vector $\mathbf{f}(\hat{\mathbf{X}})$) can be seen as an approximation of the cover image statistics."

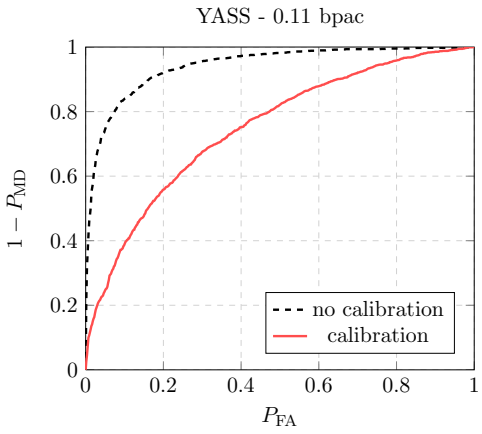


Figure 4.2.2: ROC curves for YASS using the calibrated PEV feature set and its non-calibrated version. We used YASS setting 1 which embeds 0.11 bpac on average, see Appendix A.8 for more details.

This claim was demonstrated on the histogram of coefficients from an individual DCT mode after full embedding of the F5 algorithm [132]. We reproduced this experiment and confirm the claims – the reference image histogram is, indeed, closer to the cover image histogram. The histograms are shown in Figure 4.2.1 (left).

Since 2002, however, feature-based steganalysis dramatically improved and pushed steganography to much smaller payloads. At the same time, the F5 algorithm evolved into its more secure variant, the so-called nsF5 [51], which can embed the same payload using fewer embedding changes.[3] As a result, the experiment in Figure 4.2.1 (left) is no longer relevant to today's steganalysis as such a large number of changes would be easily detectable with essentially no errors. Instead, we repeated the experiment with nsF5 at payload 0.2 bpac, which corresponds to the change rate 0.04 changes per non-zero AC DCT coefficient. This is 25 times fewer changes than with the full embedding! The resulting impact on histograms is shown in Figure 4.2.1 (right). Even though this payload is still rather large, we can see that the histogram of the reference image no longer approximates the cover image histogram. In fact, the stego image histogram is a better approximation. Quantifying this observation using the L_2 norm between histograms, for the full payload of 1.0 bpac, the reference image histogram is on average (over all images in the CAMERA database) 3.3 times closer to the cover image histogram than the stego image histogram. On the other hand, for a smaller payload of 0.2 bpac, the stego image histogram is 2.9 times closer to the cover image histogram than the reference image histogram.

We would like to point out that even though the reference image does not really approximate the cover image (or its statistics), calibration may still improve steganalysis, which goes against the intuitive explanation of calibration as provided in [44]. To demonstrate this, we performed steganalysis of nsF5 for relative payload 0.2 bpac with the 11-dimensional global histogram of DCT coefficients as the feature vector $\mathbf{f}(\mathbf{X}) = (h_{-5}(\mathbf{X}), \dots, h_5(\mathbf{X}))$. A non-calibrated feature vector leads to the error rate $P_{\mathrm{E}} = 0.46$, which is very close to random guessing. Note that by a non-calibrated version we mean $\mathbf{f}(\mathbf{X})$ instead of the difference $\mathbf{f}(\hat{\mathbf{X}}) - \mathbf{f}(\mathbf{X})$. Using a calibrated version of $\mathbf{f}$, the error is reduced to $P_{\mathrm{E}} = 0.28$.

To further motivate our study, we present the results of steganalysis of YASS when the calibration is *turned off*. The steganographic algorithm YASS was developed with the intention to disable the positive effect of calibration in steganalysis. The steganalysis results independently reported in [118, 75, 90] indicate that the feature set PEV, indeed, cannot detect YASS reliably. However, when a non-calibrated version of the PEV feature set is used, YASS becomes significantly more

[3] See Appendix A.5 for more details about nsF5 algorithm and its comparison to F5.

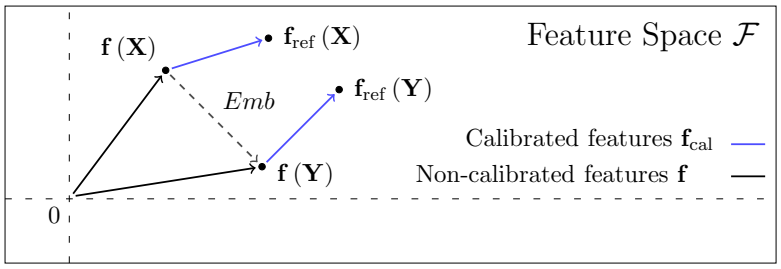


Figure 4.2.3: A diagram showing the (non-calibrated) cover- and stego-image features $\mathbf{f}(\mathbf{X})$, $\mathbf{f}(\mathbf{Y})$, with their corresponding reference features $\mathbf{f}_{\text{ref}}(\mathbf{X})$, $\mathbf{f}_{\text{ref}}(\mathbf{Y})$. Blue arrows correspond to the calibrated features.

detectable (see Figure 4.2.2). We remark here that non-calibrated feature sets other than PEV were also shown to detect YASS relatively reliably [90].

The presented experiments provoke numerous important questions. How exactly does calibration affect statistical detectability of steganographic algorithms and why does it fail for YASS? Generally, under what conditions does calibration help and when does it make steganalysis worse? What is the real purpose of calibration if it is not to approximate the cover image?

4.2.2 Mathematical framework for calibration

The process of embedding a secret message is realized by the embedding mapping $Emb : \mathcal{C} \times \mathcal{M} \times \mathcal{K} \to \mathcal{C}$ defined by (2.4.1) on page 15. For a specific message $\mathbf{m} \in \mathcal{M}$ and a stego-key $k \in \mathcal{K}$, we can drop the input parameter spaces $\mathcal{M}$ and $\mathcal{K}$, simplifying the embedding function to $Emb : \mathcal{C} \to \mathcal{C}$. We will keep the previously introduced notation and denote the cover image $\mathbf{X}$ and the corresponding stego image $\mathbf{Y} = Emb(\mathbf{X})$.

The central concept in calibration is the *reference transform* $ref : \mathcal{C} \to \mathcal{C}$, which maps the image $\mathbf{Z} \in \mathcal{C}$ to the reference image $\mathbf{Z}_{\text{ref}} \in \mathcal{C}$. One example of such a mapping ref is the spatial-domain cropping followed by compression shown in Figure 4.1.3. In steganalysis of ± 1 embedding [62, 63], the reference mapping was realized by resizing by a factor of two. The prediction filter in WS steganalysis [41] can also be interpreted as a reference transformation. We denote the feature vector of the reference image as $\mathbf{f}_{\text{ref}} = \mathbf{f} \circ ref$, where $\circ$ stands for the composition of mappings. We refer to

$$\mathbf{f}_{\text{ref}}(\mathbf{Z}) \equiv \mathbf{f}(ref(\mathbf{Z})) \equiv \mathbf{f}(\mathbf{Z}_{\text{ref}}) \tag{4.2.1}$$

as the *reference feature* of image $\mathbf{Z}$. The *calibrated feature* is defined simply as the difference between the feature vectors extracted from the image and its reference version

$$\mathbf{f}_{\text{cal}}(\mathbf{Z}) \triangleq \mathbf{f}_{\text{ref}}(\mathbf{Z}) - \mathbf{f}(\mathbf{Z}), \quad \forall \mathbf{Z} \in \mathcal{C}. \tag{4.2.2}$$

Figure 4.2.3 clarifies the introduced notation.

We now explain a framework within which the relationships among $\mathbf{f}(\mathbf{X}), \mathbf{f}(\mathbf{Y}), \mathbf{f}_{\text{ref}}(\mathbf{X})$, and $\mathbf{f}_{\text{ref}}(\mathbf{Y})$ can be formulated and quantified solely within the scope of the feature space $\mathcal{F}$. Modeling cover images as a random variable c on $\mathcal{C}$, the cover feature vector $\mathbf{f}(\mathsf{c})$ is a random variable on $\mathcal{F}$ whose central tendency and spread will be described using robust statistics, median $\mathbf{m}_{\text{c}}$ and the Median Absolute Deviation (MAD) M_{c}:

$$\mathbf{m}_{\text{c}} = \text{median}\left[\mathbf{f}(\mathsf{c})\right], \tag{4.2.3}$$

$$M_{\text{c}} = \text{median}\left[\|\mathbf{f}(\mathsf{c}) - \mathbf{m}_{\text{c}}\|_2\right], \tag{4.2.4}$$

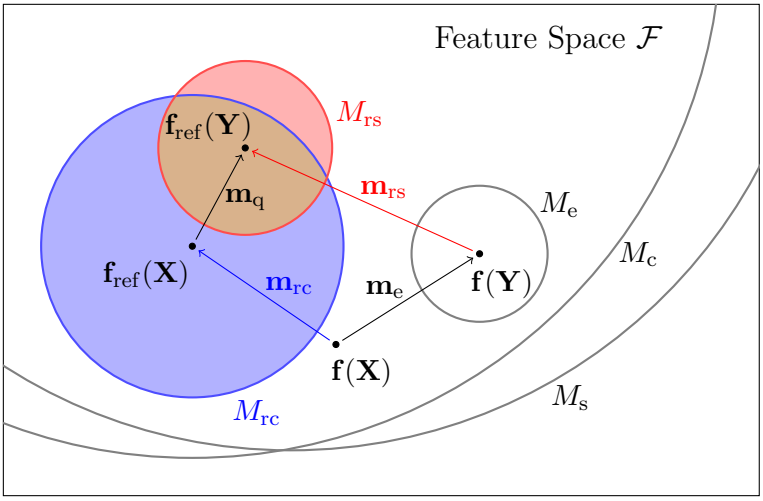


Figure 4.2.4: Two-dimensional illustration of the feature-space model as introduced in Section 4.2.2.

both calculated over all cover images in the CAMERA database. Note that M_{c} is a scalar quantity while $\mathbf{m}_{\text{c}}$ is generally a vector because the median is applied to each coordinate of $\mathbf{f}(\mathsf{c})$. The symbol $\|\cdot\|_2$ denotes the L_2 norm. The steganographic embedding, $\mathbf{Y} = Emb(\mathbf{X})$, is modeled as a shift $\mathbf{f}(\mathbf{X}) \to \mathbf{f}(\mathbf{Y})$ in the feature space represented as the difference $\mathbf{f}(\mathbf{Y}) - \mathbf{f}(\mathbf{X})$, which we again consider as a vector random variable with median $\mathbf{m}_{\text{e}}$ and MAD M_{e}. This time, the random variable is taken over covers, stego keys, and messages, all distributed uniformly on their corresponding spaces. Note that even if the embedding shift $\mathbf{f}(\mathbf{X}) \to \mathbf{f}(\mathbf{Y})$ is truly random, or even if there is no shift at all, it can still be described by $\mathbf{m}_{\text{e}}$ and M_{e}, and calibration might still work (see Example 5 in the next section, divergent reference).

Next, we consider the process of cover-image calibration as another feature space shift, $\mathbf{f}(\mathbf{X}) \to \mathbf{f}_{\text{ref}}(\mathbf{X})$, with the difference $\mathbf{f}_{\text{ref}}(\mathbf{X}) - \mathbf{f}(\mathbf{X})$ with median $\mathbf{m}_{\text{rc}}$ and MAD M_{rc} ("rc" standing for reference-cover). Similarly, we use $\mathbf{m}_{\text{rs}}$ and M_{rs} as statistical descriptors of the stego-calibration shift $\mathbf{f}(\mathbf{Y}) \to \mathbf{f}_{\text{ref}}(\mathbf{Y})$. Here, we need to keep in mind that $\mathbf{f}_{\text{ref}} = \mathbf{f} \circ ref$ and that its domain is, in fact, the original space $\mathcal{C}$. We can think of the image $\mathbf{Z}$ as a side-information for the feature space transform $\mathbf{f}(\mathbf{Z}) \to \mathbf{f}_{\text{ref}}(\mathbf{Z})$.

Finally, in some situations it might be useful to view $\mathbf{f}_{\text{ref}}(\mathbf{Y})$ with respect to $\mathbf{f}_{\text{ref}}(\mathbf{X})$, as the shift $\mathbf{f}_{\text{ref}}(\mathbf{X}) \to \mathbf{f}_{\text{ref}}(\mathbf{Y})$ with median $\mathbf{m}_{\text{q}}$ and MAD M_{q}. This, indeed, makes sense because the reference features of cover and stego images are often required to be close to each other (with the exception of Example 5).

Since a one dimensional sketch would be less informative, we illustrate the introduced concepts in two dimensions in Figure 4.2.4. The benefit of this framework is that it is laid out entirely in the feature space $\mathcal{F}$.

4.2.3 Canonical examples

Utilizing the introduced notation, we now present a series of canonical examples of how the reference feature mapping $\mathbf{f}_{\text{ref}}$ might look like and how it influences the distinguishability between the classes of cover and stego features. Our goal is to determine the properties that $\mathbf{f}_{\text{ref}}$ should possess to improve steganalysis. Note that according to the definition of $\mathbf{f}_{\text{ref}}$, it is fully determined by the feature mapping $\mathbf{f}$ and the reference transform ref. Follow Figure 4.2.5 for a schematic illustration of individual examples. For better readability, we follow the color-coding of Figures 4.2.4 and 4.2.5 in the text.

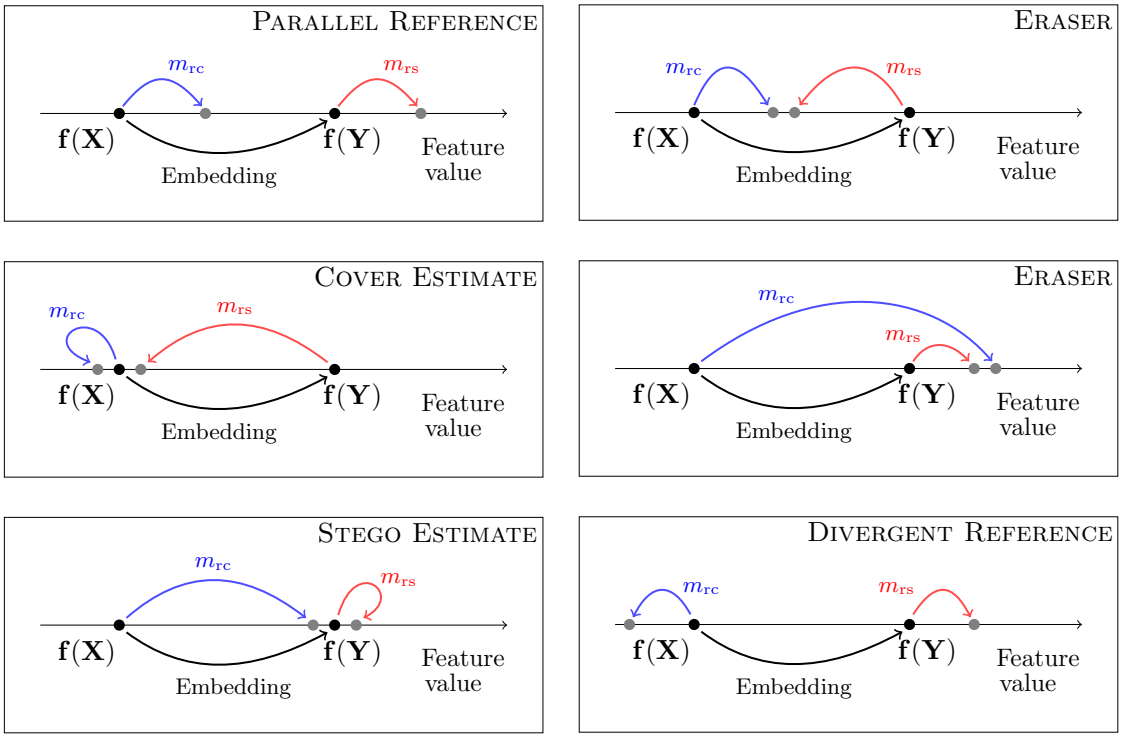


Figure 4.2.5: One-dimensional visualization of individual examples of calibration discussed in the text.

Example 1. Parallel Reference

In this first example, $\mathbf{f}_{\text{ref}}(\mathbf{X}) = \mathbf{f}(\mathbf{X}) + \mathbf{f}^\star$, where $\mathbf{f}^\star$ is some specific feature vector. In other words, calibration can be seen as a constant feature-space shift, $m_{\text{rc}} = m_{\text{rs}} = \mathbf{f}^\star$, $M_{\text{rc}} = M_{\text{rs}} = 0$. As a result, $\mathbf{f}_{\text{cal}}(\mathbf{X}) = \mathbf{f}_{\text{ref}}(\mathbf{X}) - \mathbf{f}(\mathbf{X}) = \mathbf{f}^\star$, $\forall \mathbf{X} \in \mathcal{C}$. Therefore, applying calibration causes a complete failure of steganalysis because the classes of cover and stego images become indistinguishable. We call this situation *parallel reference* since actions of ref on cover and stego images can be seen as parallel shifts in the feature space.

In practice, the shift will not be the same for every image. Nevertheless, calibration still fails if $m_{\text{rc}} \approx m_{\text{rs}}$ and $M_{\text{rc}} \approx M_{\text{rs}}$. According to our experiments, most PEV features are of this type when detecting the steganographic algorithm YASS (see the results in Section 4.2.4).

Example 2. Cover Estimate

Here, the reference transform ref maps each stego image to an image $ref(\mathbf{Y}) = \hat{\mathbf{X}}$ whose feature approximates the cover image feature while $\mathbf{f}_{\text{ref}}(\mathbf{X}) \approx \mathbf{f}(\mathbf{X})$ for the cover image. Symbolically, $\mathbf{f}_{\text{ref}}(\mathbf{Y}) = \mathbf{f}(\hat{\mathbf{X}}) \approx \mathbf{f}_{\text{ref}}(\mathbf{X}) \approx \mathbf{f}(\mathbf{X})$ and therefore $m_{\text{e}} \approx -m_{\text{rs}}$ and $M_{\text{e}} \approx M_{\text{rs}}$ with small values of m_{rc} and M_{rc} (follow Figure 4.2.4). Note that this scenario stood behind the original idea of calibration – to come up with a good cover-image estimate [44, 62, 63].

The more accurate cover estimate is provided by the reference mapping ref (the smaller are m_{rc} and M_{rc}), and the more different is stego-image feature from the cover-image feature (larger value of m_{e}), the better the benefit of calibration of this form. An interesting situation may occur in the special case of $m_{\text{e}} \approx M_{\text{e}} \approx 0$ when embedding approximately preserves the feature vector. In this case, even though the original cover/stego distinguishability is small, the cover-estimating calibration may improve detection provided $\mathbf{f}_{\text{cal}}(\mathbf{X})$ and $\mathbf{f}_{\text{cal}}(\mathbf{Y})$ exhibit different statistical properties. Note that this

is only possible because the reference mapping ref operates, in fact, in the original space of images $\mathcal{C}$, even though we model it as a feature-space shift. This situation is covered by Example 5.

Example 3. Stego Estimate

This is a complementary situation to the previous example in which ref provides an estimate of the stego feature instead of the cover feature. In other words, the values of m_{rs} and M_{rs} are small and the impact of the reference transform on cover images is similar to the impact of embedding, $m_{\text{e}} \approx m_{\text{rc}}$ and $M_{\text{e}} \approx M_{\text{rc}}$. A practical form of this example may be realized by repetitive embedding, when the feature value changes significantly when applied to the cover image, while it has a much smaller effect on stego images. This form of calibration may be especially useful for attacking idempotent embedding operations, such as LSB embedding. A real-life example of this scenario is the targeted attack on OutGuess [45].

Before we proceed with the next example, note that in this scenario (and in the previous scenario where ref provided a cover-feature estimate) the actual value of $\mathbf{f}(\mathbf{Y})$ is not important, provided it is far enough from $\mathbf{f}(\mathbf{X})$ in terms of distance in $\mathcal{F}$. Especially note that we do not require the embedding operation to shift the feature vector consistently in one direction. Provided the embedding operation indeed shifts the feature vector of the given image in the feature space consistently in one direction, we consider the next situation.

Example 4. Eraser

Here, the reference image does not provide estimates of cover- or stego-image features. Instead, we require

1. $\mathbf{f}_{\text{ref}}(\mathbf{X}) \approx \mathbf{f}_{\text{ref}}(\mathbf{Y}) \triangleq \mathbf{f}^{\star}$, the reference cover- and stego-image features should be close to each other, ideally identical.
2. $\mathbf{f}^{\star}$ should be *close enough* to both $\mathbf{f}(\mathbf{X})$ and $\mathbf{f}(\mathbf{Y})$.

Requirement 1 means that ref has to be robust w.r.t. embedding changes. Alternatively, we will say that ref *erases embedding changes* (hence the name for this scenario). In terms of the notation from Figure 4.2.4, the shift $\mathbf{f}_{\text{ref}}(\mathbf{X}) \to \mathbf{f}_{\text{ref}}(\mathbf{Y})$ should be small, $m_{\text{q}} \approx 0$, $M_{\text{q}} \approx 0$. Requirement 2 ensures that the calibration is non-trivial in the following sense. Suppose ref trivially maps all images to one specific image $\mathbf{Z} \in \mathcal{C}$. Consequently, $\mathbf{f}_{\text{ref}}(\mathbf{Z}') = \mathbf{f}(\mathbf{Z}) = const.$, $\forall \mathbf{Z}' \in \mathcal{C}$. Even though the first requirement is ideally satisfied, the calibrated features $\mathbf{f}_{\text{cal}}$ defined by (4.2.2) are just shifted (and negative) versions of the original features $\mathbf{f}$ and calibration has no effect on the distinguishability between $\mathbf{X}$ and $\mathbf{Y}$. Therefore, $\mathbf{f}^{\star}$ should be close to both $\mathbf{f}(\mathbf{X})$ and $\mathbf{f}(\mathbf{Y})$. Furthermore, the closer we are with $\mathbf{f}^{\star}$ to the original cover- and stego-image features, the smaller the variance of $\mathbf{f}^{\star}$ is and the better detection we can expect.

We stress that in this case the requirement of $\mathbf{f}(\mathbf{Y})$ being different enough from $\mathbf{f}(\mathbf{X})$ is not sufficient. In order to make calibration work here, we require the embedding shift $\mathbf{f}(\mathbf{X}) \to \mathbf{f}(\mathbf{Y})$ to be consistent in terms of direction.

Example 5. Divergent Reference

By divergent reference, we mean the situation when $\mathbf{f}_{\text{ref}}(\mathbf{X}) = \mathbf{f}(\mathbf{X}) - \mathbf{f}_1$, $\mathbf{f}_{\text{ref}}(\mathbf{Y}) = \mathbf{f}(\mathbf{Y}) - \mathbf{f}_2$, $\mathbf{f}_1 \neq \mathbf{f}_2$. In other words, the action of the reference mapping can be interpreted as a feature space shift of the original feature vector $\mathbf{f}(\mathbf{Z})$ to a different direction depending on whether the input $\mathbf{Z}$ is a cover or a stego image, $m_{\text{rc}} = \mathbf{f}_1$, $m_{\text{rs}} = \mathbf{f}_2$, $M_{\text{rc}} = M_{\text{rs}} = 0$. Therefore, the resulting calibrated feature will be $\mathbf{f}_1$ for the cover image and $\mathbf{f}_2$ for the stego image, implying perfect detectability. In practice, it is not possible to achieve exactly the same shift for every cover and stego image as in this case $\mathbf{f}_{\text{cal}}$ would basically serve as a detector itself, returning the label $\mathbf{f}_1$ for cover and $\mathbf{f}_2$ for stego. Instead, we simply require the statistics $m_{\text{rc}}, M_{\text{rc}}$ and $m_{\text{rs}}, M_{\text{rs}}$ to be different.

Remarkably, in this example calibration will work even when the steganography preserves the feature vector, $m_e = M_e = 0$, because ref takes as the input the whole image $\mathbf{Z} \in \mathcal{C}$ and not just its feature vector! A good example of this scenario is the situation when we use the histogram bins of zeros and ones to attack Jsteg. Because Jsteg preserves the counts of zeros and ones, $\mathbf{f}(\mathbf{X}) = \mathbf{f}(\mathbf{Y})$, the features themselves are useless for steganalysis of Jsteg. However, their calibrated versions improve the distinguishability between cover and stego features because the reference mapping ref reacts differently to cover and stego images (see Cases 5.1 and 5.2 in Section 4.2.4).

4.2.4 Validation of the framework

In this section, we justify the usefulness of the proposed mathematical framework, illustrated by Figure 4.2.4, for studying the effects of calibration on real steganographic techniques. In particular, we consider the following steganographic schemes: nsF5 [51], the MME algorithm [72], Jsteg [129], JPHide&Seek, Steghide [57], and YASS [126, 118]. All listed algorithms are described in Appendix A. We use the CAMERA image database and study the impacts of calibration on the individual features of the PEV feature set, see Table 4.1. For each steganographic method and relative payload, we obtained 6500 cover images and the same number of stego images. For both cover and stego images, their corresponding reference images were created using the spatial-domain cropping as explained in Figure 4.1.3. The non-calibrated PEV features were extracted from all cover and stego images and their corresponding reference versions. All features were scaled so that cover-image features exhibited unit variance. Finally, the values of $m_e, M_e, m_q, M_q, m_{rc}, M_{rc}, m_{rs}$, and M_{rs}, were computed separately for every feature. Here, we use non-boldface symbols because the medians will be always computed for scalar quantities.

Using the measured statistical quantities, in Table 4.2 we demonstrate on carefully selected combinations of steganographic techniques, payloads, and features that the canonical examples explained in Section 4.2.3, indeed, occur within the PEV feature set. Every case listed in Table 4.2 was given a unique index (the last column) that will be used for referencing. For better readability, the most relevant values for each case are highlighted.

The situation when the distributions of shifts $\mathbf{f}(\mathbf{X}) \to \mathbf{f}_{\text{ref}}(\mathbf{X})$ and $\mathbf{f}(\mathbf{Y}) \to \mathbf{f}_{\text{ref}}(\mathbf{Y})$ are very similar, and therefore calibration hurts performance, was called Parallel Reference in Section 4.2.3. Cases 1.1–1.6 show examples of Parallel Reference features because $m_{rc} \approx m_{rs}$ and $M_{rc} \approx M_{rs}$. Cases 1.1–1.3 correspond to the YASS algorithm, which exhibited the most frequent occurrence of this effect in our tests. Parallel Reference, however, may occur for some features for other algorithms as well (Cases 1.4–1.6).

The Cover Estimate Example 2 from Section 4.2.3 describes the situation when the reference transform improves steganalysis by making calibrated features of cover images approximately zero and calibrated features of stego images nonzero. This example is characterized by $m_e \approx -m_{rs}$ and $M_e \approx M_{rs}$ with small values of m_{rc} and M_{rc}. This situation can be nicely observed for embedding-sensitive features and large payloads, where the reference transform ref indeed provides an estimate of the cover feature. Cases 2.1–2.3 in Table 4.2 correspond to features that significantly change with embedding (histogram bins for nsF5 and Jsteg and co-occurrence C_{11} for nsF5). The histogram of the DCT mode $(2, 1)$ (Case 2.4) also falls into this category as Steghide preserves only the global histogram but not necessarily the histograms of individual DCT modes.

With decreasing payload size, Cover Estimate is less likely to occur because the embedding distortion becomes smaller while the properties of the reference mapping remain unchanged.

Within the PEV feature set, we did not observe any cases of Example 3, the Stego Image Estimation, for any steganographic scheme. A real-life example of this scenario is the attack on OutGuess [45].

We now proceed to Example 4 called Eraser. In Table 4.2, we demonstrate this by Cases 4.1–4.3. The characterizing property of this scenario is that the reference features are close to each other when compared with the size of a consistent embedding shift. In other words, the median and MAD of

Algorithm (bpac)	Feature	m_{e}	M_{e}	m_{q}	M_{q}	m_{rc}	M_{rc}	m_{rs}	M_{rs}	Example.Case
YASS 3 (0.19)	v	+0.015	0.005	+0.015	0.005	+0.011	0.010	+0.011	0.010	1.1
YASS 2 (0.05)	b_2	+0.012	0.006	+0.012	0.007	−0.005	0.027	−0.005	0.027	1.2
YASS 1 (0.11)	M_{-4-4}	+0.001	0.014	+0.001	0.016	+0.008	0.045	+0.008	0.045	1.3
nsF5 (0.20)	$d_{-1}^{(4,1)}$	+0.015	0.017	+0.012	0.034	+0.043	0.104	+0.043	0.104	1.4
MME (0.10)	M_{0-3}	−0.001	0.011	−0.000	0.016	+0.014	0.031	+0.015	0.031	1.5
JPHS (0.10)	$a_{-2}^{(1,3)}$	+0.000	0.000	+0.000	0.000	+0.059	0.144	+0.059	0.144	1.6
nsF5 (1.00)	h_0	+0.535	0.145	+0.170	0.073	−0.019	0.020	−0.381	0.082	2.1
nsF5 (1.00)	C_{11}	−1.464	0.333	−0.163	0.103	+0.064	0.072	+1.373	0.258	2.2
Jsteg (0.20)	h_{-2}	+0.528	0.142	+0.060	0.023	+0.015	0.028	−0.450	0.128	2.3
Steghide (0.20)	$a_2^{(2,1)}$	−0.227	0.108	+0.030	0.081	+0.007	0.154	+0.252	0.178	2.4
nsF5 (0.20)	$a_0^{(1,2)}$	+0.043	0.010	−0.002	0.009	−0.071	0.034	−0.118	0.038	4.1
MME (0.10)	M_{00}	+0.017	0.003	+0.002	0.004	−0.020	0.023	−0.035	0.023	4.2
MME (0.10)	h_0	+0.015	0.002	+0.002	0.003	−0.019	0.020	−0.032	0.020	4.3
JPHS (0.10)	$a_{-1}^{(1,2)}$	+0.000	0.000	+0.437	0.399	+0.108	0.134	+0.563	0.488	5.1
Jsteg (0.20)	h_1	+0.000	0.000	+0.113	0.023	+0.024	0.039	+0.131	0.047	5.2
Steghide (0.20)	C_{-11}	−0.005	0.011	+0.122	0.030	+0.059	0.046	+0.183	0.061	5.3
nsF5 (0.20)	M_{-13}	+0.001	0.026	−0.024	0.049	+0.015	0.110	−0.010	0.107	5.4
YASS 4 (0.12)	M_{31}	−0.286	0.265	−0.175	0.285	+0.118	0.326	+0.242	0.326	5.5

Table 4.2: Experimental verification of calibration examples from Section 4.2.2. For selected combinations of the embedding method, payload, and PEV feature (notation taken from Table 4.1), we computed the sample statistics $m_{\mathrm{e}}, M_{\mathrm{e}}, m_{\mathrm{q}}, M_{\mathrm{q}}, m_{\mathrm{rc}}, M_{\mathrm{rc}}, m_{\mathrm{rs}}$, and M_{rs}. For better readability, values most relevant to individual cases are highlighted.

the shift $\mathbf{f}_{\mathrm{ref}}(\mathbf{X}) \to \mathbf{f}_{\mathrm{ref}}(\mathbf{Y})$ should be small, $m_{\mathrm{q}} \approx 0$, $M_{\mathrm{q}} \approx 0$, despite the rather large relative values of distortions caused by the reference mapping (large values of $m_{\mathrm{rc}}, M_{\mathrm{rc}}$ and $m_{\mathrm{rs}}, M_{\mathrm{rs}}$). Note that for small payloads, the histogram bin of a steganographic scheme that disturbs first-order statistics may become an Eraser rather than Cover Estimation (Case 4.1).

By far the most frequent situation was the Divergent Reference illustrated by the last set of Cases 5.1–5.5. Here, as opposed to the Parallel Reference (Cases 1.1–1.6), the reference statistics $m_{\mathrm{rc}}, M_{\mathrm{rc}}$ and $m_{\mathrm{rs}}, M_{\mathrm{rs}}$ should simply be different. The more different they are, the larger the benefit of calibration. Cases 5.1 and 5.2 demonstrate the intriguing situation when the feature value itself is preserved during embedding (and therefore useless for steganalysis), while its calibrated form has a good distinguishing power due to the fact that the reference transform r reacts differently to cover and stego images. Cases 5.3–5.5 were included to illustrate Divergent Reference on various steganographic methods and non-preserved features.

Note that all cases, with the exception of Parallel Reference, can be interpreted as special cases of Divergent Reference. Since both Cover Estimate and Eraser need the existence of the embedding shift $\mathbf{f}(\mathbf{X}) \to \mathbf{f}(\mathbf{Y})$, the property of the reference transform, $\mathbf{f}_{\mathrm{ref}}(\mathbf{X}) \approx \mathbf{f}_{\mathrm{ref}}(\mathbf{Y})$, basically implies that the shifts $\mathbf{f}(\mathbf{X}) \to \mathbf{f}_{\mathrm{ref}}(\mathbf{X})$ and $\mathbf{f}(\mathbf{Y}) \to \mathbf{f}_{\mathrm{ref}}(\mathbf{Y})$ must be different. This is what we request from calibration – the two shifts must be as different as possible in order to easily distinguish between cover and stego features.

To summarize our observations, we showed that calibration does not have to (and in general it does not) approximate the cover-image feature to improve steganalysis. At the same time, we showed examples when calibration is harmful (Cases 1.1–1.5). Moreover, all five calibration scenarios

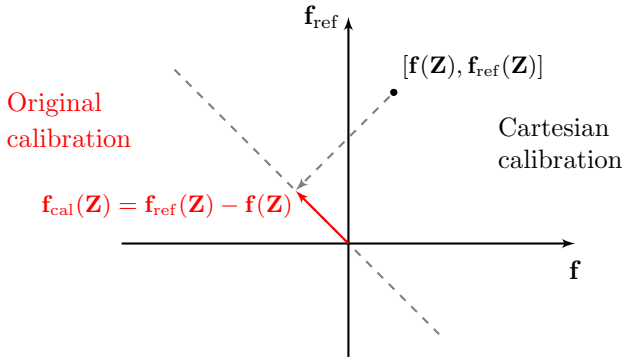


Figure 4.2.6: Cartesian calibration and its relationship to the original calibration.

described in Section 4.2.3 do occur in real life.

An important fact that needs to be stressed is that we only studied each feature individually while ignoring the dependencies among individual features. Therefore, we have to be careful about the interpretation in terms of the *global behavior* within the PEV feature set. The individual features may be useful even without calibration, i.e., without their baseline value provided by the reference mapping ref, when we utilize dependencies among them. This topic is the subject of the next section.

4.2.5 Cartesian calibration

The experiments described so far demonstrate that several different mechanisms are responsible for the positive effect of calibration. At the same time, calibration may have a catastrophically negative effect on steganalysis when Parallel Reference occurs. To prevent such failures, in this section we propose a modified calibration procedure and demonstrate that it improves steganalysis in practice.

Let us assume that $\mathbf{f}$ is a d-dimensional feature mapping. Given an image $\mathbf{Z} \in \mathcal{C}$, we extract its feature vector $\mathbf{f}(\mathbf{Z})$ and the reference feature vector $\mathbf{f}_{\text{ref}}(\mathbf{Z})$. Applying a linear transformation to the $2d$-dimensional vector $(\mathbf{f}_{\text{ref}}(\mathbf{Z}), \mathbf{f}(\mathbf{Z}))$, we obtain the vector $\hat{\mathbf{f}}(\mathbf{Z}) = (\mathbf{f}_{\text{ref}}(\mathbf{Z}) - \mathbf{f}(\mathbf{Z}),\ \mathbf{f}_{\text{ref}}(\mathbf{Z}) + \mathbf{f}(\mathbf{Z})) = (\mathbf{f}_{\text{cal}}(\mathbf{Z}), \mathbf{f}_{\text{ref}}(\mathbf{Z}) + \mathbf{f}(\mathbf{Z}))$. Since the first half of $\hat{\mathbf{f}}$ is the calibrated feature vector $\mathbf{f}_{\text{cal}}$, we can think of calibration as a d-dimensional projection of $\hat{\mathbf{f}}$ to its first half. This projection, however, may remove potentially useful information contained in the second half of the feature vector $\hat{\mathbf{f}}$.

Therefore, we propose to calibrate by taking the Cartesian product of features and their reference values, rather than their differences. The Cartesian calibration has several advantages over the original difference-based calibration:

- The difference may not be the best way of utilizing the information contained in the reference values. The Cartesian calibration leaves it on the machine learning tool to decide.
- Even when the reference feature vector $\mathbf{f}_{\text{ref}}$ is useless as far as the distinguishability between cover and stego classes is concerned (Parallel Reference), the performance of the steganalyzer will not be compromised. In that case, $\mathbf{f}_{\text{ref}}$ will simply be a non-influential part of the feature vector.
- As most of the features are usually correlated, an important aspect of the Cartesian calibration is that the individual elements of the vector $\mathbf{f}_{\text{ref}}$ can serve as references to many different features from $\mathbf{f}$, not only to those that the difference-based calibration subtracts them from.

The only disadvantage of the Cartesian calibration is the increased feature space dimensionality, and therefore care needs to be taken when choosing a machine-learning tool.

Figure 4.2.6 illustrates the relationship of the Cartesian calibration to the original way of calibration by difference.

We subjected the Cartesian calibration to a large-scale test. For each embedding method and payload, we constructed a separate steganalyzer implemented as the Gaussian SVM[4] using the following three feature sets:

1. PEV – 274-dimensional feature set as proposed in [107], i.e. calibrated by subtracting $\mathbf{f}_{\text{ref}}$,
2. NC-PEV – the non-calibrated version of PEV feature set, dimensionality = 274,
3. CC-PEV – Cartesian-calibrated PEV feature set of dimensionality $2 \times 274 = 548$.

Since our goal was to compare the performance of individual feature sets rather than embedding methods, we chose different payloads for different methods, depending on the detectability. (We wanted payloads for which the methods would be neither too easy nor too difficult to detect.) For the steganographic methods nsF5, MME, and JPHS, we chose the payloads 0.05,0.10,0.15, and 0.20 bpac. For more detectable algorithms, Jsteg and Steghide, we chose smaller payloads, 0.02,0.03,0.04, and 0.05 bpac. For YASS, since the payload cannot be easily controlled, we show averages over all images in our database. The results are summarized in Table 4.3.

The error rates P_{E} were obtained for five different divisions of the database into a training and a testing set. In Table 4.3, we report the average values. The CC-PEV feature set always produces the best steganalysis. Note that the difference-based calibration involved in PEV makes steganalysis of YASS remarkably worse than if no calibration was used (NC-PEV), for all settings. This is not surprising because YASS was created with the intention to disable calibration. On the other hand, calibration by Cartesian product (CC-PEV) improves the detectability of YASS, compared to NC-PEV features. This means that there are features for which even for YASS the reference mapping ref improves steganalysis.

A careful inspection of the table reveals that except for JPHide&Seek, where the calibrated features PEV perform significantly better than the non-calibrated features NC-PEV, and YASS, where the PEV features failed, the feature sets PEV and NC-PEV have a very similar performance. This is rather surprising because calibration was thought to improve steganalysis.

We provide the following heuristic explanation for this phenomenon. The key observation is that the individual features involved in NC-PEV exhibit strong dependencies. If we put two correlated features next to each other, they serve mutually as "reference" values in the same sense as if we put $\mathbf{f}$ and $\mathbf{f}_{\text{ref}}$ next to each other as in our modified calibration procedure. Consequently, the steganalysis performance of two dependent features may be remarkably better than if we used those features individually (as we did in our experiments in Section 4.2.4). Taking this to an extreme, we can say that not only pairs of features but also the individual elements of the entire feature vector mutually "calibrate" each other, and even the features that perform poorly *individually* may be very useful when added as references to other features.

4.2.6 Concluding remarks

In the past, calibration has been proposed as a process in which a steganalysis feature is supplied with a baseline (reference) value to improve the feature's ability to distinguish between cover and stego features. Typically, calibration has been carried out as a difference between the original feature value and its reference value. However, even though calibration is generally recognized as beneficial, we

[4]The hyper-parameters of each SVM were optimized through the process of 5-fold cross-validation and the grid-search over a pre-defined grid of values.

		Error P_E		
Algorithm	bpac	NC-PEV	PEV	CC-PEV
nsF5	0.05	0.361	0.360	**0.331**
	0.10	0.202	0.218	**0.177**
	0.15	0.100	0.094	**0.077**
	0.20	0.048	0.040	**0.036**
Jsteg	0.02	0.097	0.132	**0.083**
	0.03	0.042	0.051	**0.032**
	0.04	0.022	0.021	**0.018**
	0.05	0.015	0.013	**0.010**
Steghide	0.02	0.114	0.127	**0.083**
	0.03	0.055	0.056	**0.043**
	0.04	0.031	0.031	**0.024**
	0.05	0.021	0.015	**0.011**
MME	0.05	0.309	0.310	**0.277**
	0.10	0.187	0.207	**0.165**
	0.15	0.130	0.149	**0.107**
	0.20	0.023	0.017	**0.012**
JPHS	0.05	0.306	0.100	**0.094**
	0.10	0.160	0.066	**0.054**
	0.15	0.076	0.034	**0.022**
	0.20	0.039	0.014	**0.006**
YASS 1	0.110	0.133	0.317	**0.113**
YASS 2	0.051	0.179	0.347	**0.164**
YASS 3	0.187	0.102	0.121	**0.082**
YASS 4	0.118	0.120	0.303	**0.109**
YASS 5	0.159	0.075	0.241	**0.064**
YASS 6	0.032	0.269	0.342	**0.258**
YASS 7	0.078	0.244	0.298	**0.225**
YASS 8	0.138	0.211	0.251	**0.180**

Table 4.3: Steganalysis of selected algorithms when using differently calibrated feature sets. CC-PEV delivers the best performance across all algorithms and payloads.

have identified situations when calibration has very little or no effect on the steganalysis performance, and also found situations where such an operation dramatically decreases the detection accuracy.

The established thesis that calibration provides an estimate of cover image features is not necessarily correct. In fact, we recognized five different archetypes of calibration based on its mechanism through which it provides a given feature with its reference value. Our view is supported by experiments on real steganographic schemes and with a feature set that is widely used for steganalysis of JPEG images. Furthermore, our newly acquired insight enabled us to propose a modified approach to calibration, the Cartesian calibration, in which the reference feature value is adopted as an additional feature instead of subtracted from the original feature value. Calibration performed in this way removes the problem of catastrophic failures for some steganographic schemes and improves steganalysis across a wide range of steganographic schemes and payloads.

In Chapter 6, Cartesian calibration will be utilized for construction of a rich model for steganalysis of JPEG images.

4.3 Dangers of incomplete models

With the increasing level of sophistication of feature spaces for steganalysis, there appears a simple recipe how to construct secure steganographic systems – make the embedding (approximately) preserve the feature-space representation of the cover image. While this strategy sounds appealing, in this section we will show that even a high-dimensional cover model does not guarantee immunity to simple attacks and that the security of steganographic techniques adopting imperfect models can be easily compromised. We will demonstrate our arguments on four specific examples of steganographic algorithms.

4.3.1 Simple statistical restoration – OutGuess

In 2001, Provos proposed the JPEG domain steganographic algorithm OutGuess [113] that adopts the histogram of DCT coefficients as a cover model. OutGuess works as follows. DCT coefficients of the cover image are divided into two disjoint parts and only the first part is used for data hiding – the least significant bits (LSBs) of the visited coefficients are replaced with the message bits. As the embedding disturbs the adopted model (the histogram of DCT coefficients), the second part of the image is used to restore the histogram to its original form. This strategy is called *statistical restoration [127]* and in this case completely preserves the DCT coefficient histogram. However, we all know that a simple histogram is a poor cover model of real JPEG images as there are various dependencies among the DCT coefficients. Therefore, even though completely secure within the first-order model, OutGuess has been successfully attacked by higher-order statistics [43, 135, 71, 22].

4.3.2 Feature-Correction Method

In 2008, we generalized OutGuess and proposed the Feature-Correction Method (FCM) [76], a general framework for embedding while approximately preserving the entire feature vector.[5] Similar to OutGuess, FCM also divides the image into two disjoint parts and uses the first part for embedding and the second for statistical restoration. Unlike OutGuess, however, FCM minimizes distortion already in the first, embedding phase. Whenever the parity (LSB) of the DCT coefficient is to be changed, the value is increased or decreased by 1 based on the corresponding feature-space distortion. Furthermore, the number of embedding changes is minimized by wet paper codes (WPC) [47].

In the second, correction phase, additional modifications are made in the unused part of the image to bring the feature vector closer to its original position. Each non-zero coefficient is visited and its value is changed by $-2, -1, +1, +2$, or left unchanged, depending on the position bringing it the closest to the original cover feature vector. Changes by more than 2 are not allowed as they would introduce visible (and thus detectable) distortion.

To complete the description of the FCM embedding algorithm, we need to supply the metric for measuring the distance between feature vectors $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{y} = (y_1, \ldots, y_n)$ in the feature space $\mathcal{F}$. In [76], we used the weighted Euclidean norm

$$d_{\mathrm{FCM}}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{\sigma_i^2}, \tag{4.3.1}$$

where σ_i^2 is the variance of the ith feature estimated from a large database of cover images.

The FCM was implemented for a specific example of a complex feature space – the 274-dimensional PEV features (calibrated by difference) discussed in the previous section. The complete list of the PEV features appears in Table 4.1. In spite of the complexity of the PEV feature space, the FCM was able to preserve it quite well. This is demonstrated in Figure 4.3.1, where we show the steganalysis

[5] FCM was later further investigated in [23].

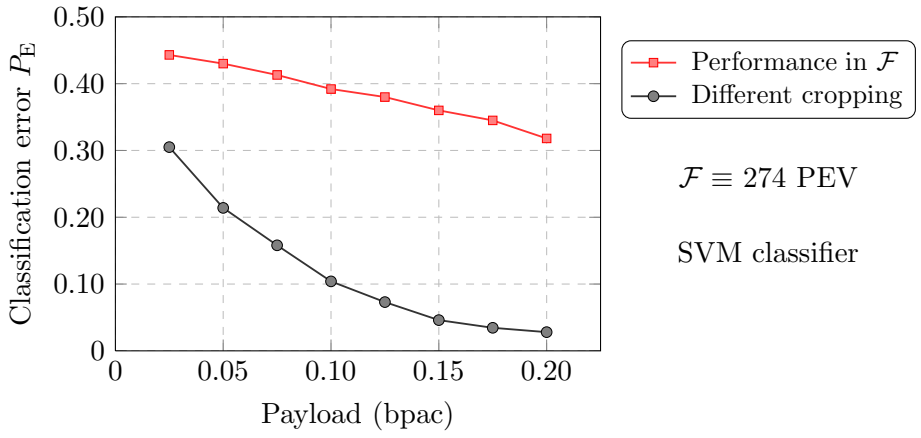


Figure 4.3.1: Steganalysis of the FCM algorithm that approximately preserves the PEV feature space $\mathcal{F}$. Red: steganalysis within the PEV model. Black: steganalysis using a slightly modified model (details in the text). We used the CAMERA image database and the non-linear SVM classifier with the Gaussian kernel.

detection performance, in terms of the classification error P_E defined by (2.6.3), as a function of the relative message length. For the purpose of this experiment, we used the CAMERA image database and utilized the SVM classifier with the Gaussian kernel whose hyper-parameters were optimized through the cross-validation. The red curve shows the achieved error rates when the PEV features are used for steganalysis, i.e., the same space that was utilized by the FCM. The error rates are quite high, suggesting a high level of security within this model. However, if we slightly modify the feature space, in this case if we replace the 4×4 cropping by 2×2,[6] the security of the FCM is compromised (black color).

To conclude, even though the FCM made embedding reasonably secure within the scope of the PEV model, only a small modification of this feature space was sufficient for a successful attack. The steganography is "overtrained" to a cover model that is not a complete statistical descriptor of the cover source.

4.3.3 Using high-dimensional models – HUGO

The spatial domain steganographic algorithm HUGO (Highly Undetectable steGO) [105], which was used in the steganalysis contest BOSS (Break Our Steganographic System) conceived as Alice's challenge to Eve, can be interpreted as a more advanced version of the FCM. The authors removed the feature correction phase and used a cover model of dimensionality more than 10^7 to make the model "more complete" and make it hopefully impossible for Eve to work outside the model. Despite the high dimension of the HUGO's model, the model contains a security flaw that could be utilized by steganalysis. In order to understand the source of the problem, we now cover essential parts of the HUGO embedding algorithm, referring the reader to the original publication for more details.

Starting with a cover image $\mathbf{X} = (X_{ij}) \in \{0, \dots, 255\}^{n_1 \times n_2}$, HUGO represents it with a feature vector computed from four three-dimensional co-occurrence matrices obtained from differences of horizontally, vertically, diagonally, and minor-diagonally neighboring pairs of pixels. The horizontal co-occurrence matrix computed from $\mathbf{X}$ is denoted $\mathbf{C}^{\rightarrow}(\mathbf{X})$ and its elements $C^{\rightarrow}_{\mathbf{d}}$, $\mathbf{d} = (d_1, d_2, d_3) \in$

[6] The original PEV feature space involves calibration with spatial domain image cropping by 4×4 pixels. Here, we replaced this cropping with the cropping by 2×2 pixels only.

$\mathcal{T}_3 = \{-T, \dots, T\}^3$, are defined as

$$C^{\rightarrow}_{\mathbf{d}}(\mathbf{X}) = \frac{1}{Z}\left|\{(D^{\rightarrow}_{ij}, D^{\rightarrow}_{i,j+1}, D^{\rightarrow}_{i,j+2}) \mid D^{\rightarrow}_{i,j+k-1} = d_k, k = 1, \dots, 3\}\right|, \tag{4.3.2}$$

where $\mathbf{D}^{\rightarrow} = (D^{\rightarrow}_{ij})$ is the difference array with elements $D^{\rightarrow}_{ij} = \mathrm{trunc}_T(X_{ij} - X_{i,j+1})$, $i = 1, \dots, n_1$, $j = 1, \dots, n_2 - 2$, $\mathrm{trunc}_T(x)$ is the truncation operator (1.5.1), and Z is the normalization factor ensuring that $\sum_{\mathbf{d}\in\mathcal{T}_3} C^{\rightarrow}_{\mathbf{d}}(\mathbf{X}) = 1$. The vertical, diagonal, and minor-diagonal matrices are defined similarly, resulting in 8 different co-occurrence matrices $\mathbf{C}^k$, $k \in \{\rightarrow, \leftarrow, \uparrow, \downarrow, \searrow, \nwarrow, \swarrow, \nearrow\}$. The final feature vector is defined as

$$f_{\mathrm{HUGO}}(\mathbf{X}) = \left\{ \left(\sum_{k\in\{\rightarrow,\leftarrow,\uparrow,\downarrow\}} C^k_{\mathbf{d}}(\mathbf{X}), \sum_{k\in\{\searrow,\nwarrow,\swarrow,\nearrow\}} C^k_{\mathbf{d}}(\mathbf{X}) \right) \middle| \mathbf{d} \in \mathcal{T}_3 \right\} \in \mathbb{R}^{2(2T+1)^3}. \tag{4.3.3}$$

The secret message is embedded by modifying pixels by ± 1 while minimizing the heuristically constructed distortion function

$$d_{\mathrm{HUGO}}(\mathbf{X}, \mathbf{Y}) = \sum_{d_1,d_2,d_3=-T}^{T} \frac{1}{(||\mathbf{d}||_2 + \sigma)^\gamma} \left| f_{\mathrm{HUGO}}(\mathbf{X}) - f_{\mathrm{HUGO}}(\mathbf{Y}) \right|, \tag{4.3.4}$$

where σ and γ are adjustable parameters. The embedding is performed using syndrome-trellis codes [36]. The default setting for the threshold T used in [105] was $T = 90$, which means that the embedding approximately preserves a $2(2T + 1)^3 = 11,859,482$-dimensional feature vector. HUGO designers have likely opted for such a high dimension to make it as hard as possible for Eve to mount an attack. Indeed, the individual co-occurrence bins with $|\mathbf{d}| > 90$ are mostly empty or very sparsely populated and the steganalyst cannot use them to make any reliable inference about the presence of a secret message. However, this does not mean that the *marginals* of the feature vector (4.3.3) will necessarily be sparsely populated as well.

For image $\mathbf{X}$, let us define the vector $\mathbf{h}^{\mathbf{X}} = (h^{\mathbf{X}}_i)$, where $h^{\mathbf{X}}_i$, $i = 0, \dots, 255$, is the total number of pixel pairs adjacent either in the horizontal, vertical, diagonal, or minor-diagonal direction whose difference in absolute value is equal to i.[7] Because pixel pairs with differences below 90 are treated differently by the embedding algorithm than pairs with differences above 90, $\mathbf{h}^{\mathbf{X}}$ contains a detectable artifact around the value of 90, where HUGO's model ends. This is confirmed in Figure 4.3.2 (left) where we show the histogram bins of cover and stego images (HUGO with payload 0.4 bpp) averaged over all BOSSbase images. Note that the bins $h^{\mathbf{X}}_{89}$ and $h^{\mathbf{X}}_{90}$ decrease while $h^{\mathbf{X}}_{91}$ and $h^{\mathbf{X}}_{92}$ increase after embedding. This is because the difference 90 is more likely to be changed to 91 than to 89 as this change goes "out of the model" and thus is less costly than the change to 89, in terms of the distortion (4.3.4). The bin 91 thus increases while bin 89 decreases. Similarly, the algorithm changes 91 to 92 as it stays out of the model while changing it to 90 would introduce a non-zero cost. Consequently, the bin 92 increases and the bin 90 decreases.

Taking the four features $(h^{\mathbf{X}}_{89}, h^{\mathbf{X}}_{90}, h^{\mathbf{X}}_{91}, h^{\mathbf{X}}_{92})$ as a feature space for steganalysis attack, in Figure 4.3.2 (right) we show the detection accuracy P_{E} for six relative payloads $\alpha \in \{0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$bpp obtained by training the Gaussian SVM. Note that the error is nearly constant w.r.t. the payload, which is highly unusual for a detection statistic. We explain this peculiar behavior by HUGO's adaptive embedding mechanism – smaller payloads introduce a higher percentage of changes in textured areas and around edges where the feature vector $\mathbf{h}^{\mathbf{X}}$ is effective. We note that the discovered security flaw in the HUGO's model (4.3.3) could be easily fixed by setting the threshold $T = 255$ which would turn the performance of the 4-dimensional histogram feature vector $(h^{\mathbf{X}}_{89}, h^{\mathbf{X}}_{90}, h^{\mathbf{X}}_{91}, h^{\mathbf{X}}_{92})$ essentially into random guessing [80].

We showed that the high dimensionality of the model, in this case more than 10^7, is not sufficient to make the steganography secure. Care needs to be taken at the boundaries of the model in order to avoid unfavorable artifacts that could be utilized by Eve.

[7] Vector $\mathbf{h}^{\mathbf{X}}$ is thus a histogram of the absolute differences of neighboring coefficients in the image $\mathbf{X}$.

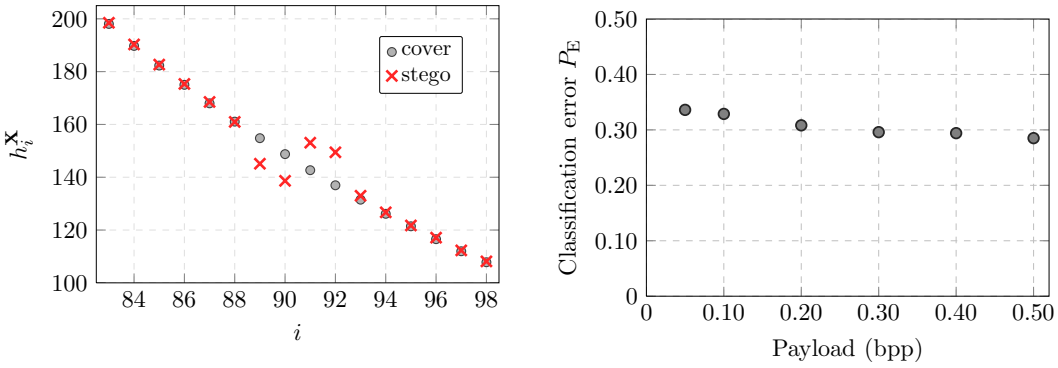


Figure 4.3.2: Left: histogram bins $h_i^{\mathbf{X}}$, $i = 83, \ldots, 98$ for cover images and for stego images embedded with HUGO with payload 0.4 bpp averaged over all BOSSbase images. Right: Steganalysis of HUGO across different payloads using 4 features $(h_{89}^{\mathbf{X}}, h_{90}^{\mathbf{X}}, h_{91}^{\mathbf{X}}, h_{92}^{\mathbf{X}})$ and the Gaussian SVM.

4.3.4 Model-optimized distortion function in JPEG domain

As the security of stegosystems built from the principle of minimum impact depends primarily on how well the distortion actually measures the statistical detectability, the authors of [35] proposed to *learn* the distortion function (its parameters) from a sample of cover and stego image features, and thus replaced the heuristic parameter determination present in both FCM and HUGO. The authors formulated the task of finding the best parameters of the distortion function as an optimization problem with the objective of minimizing the margin of a linear support vector machine (L-SVM) in a given feature space $\mathcal{F}$.

This novel and quite general framework for distortion optimization was used to construct a new JPEG steganography algorithm, which we will refer to as MOD (Model Optimized Distortion). The authors used the 548-dimensional CC-PEV feature space [77][8] chosen as a reasonable representative of a current state-of-the-art steganography model. To show that the embedding was not overtrained to this model, the authors tested MOD with the CC-PEV set with a slightly different cropping in calibration[9] as well as with the Cross-Domain Feature set (CDF) obtained by merging CC-PEV and the 686-dimensional SPAM vector [104] computed from images represented in the spatial-domain. No signs of overtraining were revealed and MOD was reported to be significantly more secure than the nsF5 algorithm [51].

We subjected the MOD algorithm to a deeper analysis and further investigation. In the algorithm, the cost ρ_{ij} of changing a DCT coefficient X_{ij} by ± 1 is determined by its immediate intra- and inter-block neighborhood:

$$\mathcal{N}_{\mathrm{ir}} = \{X_{i+8,j}, X_{i,j+8}, X_{i-8,j}, X_{i,j-8}\}, \tag{4.3.5}$$
$$N_{\mathrm{ia}} = \{X_{i+1,j}, X_{i,j+1}, X_{i-1,j}, X_{i,j-1}\}. \tag{4.3.6}$$

It is determined as a sum

$$\rho_{ij} = \sum_{z \in \mathcal{N}_{\mathrm{ia}}} (\theta^{(\mathrm{ia})}_{X_{ij}-z})^2 + \sum_{z \in \mathcal{N}_{\mathrm{ir}}} (\theta^{(\mathrm{ir})}_{X_{ij}-z})^2, \tag{4.3.7}$$

where $\boldsymbol{\theta} = (\boldsymbol{\theta}^{(\mathrm{ir})}, \boldsymbol{\theta}^{(\mathrm{ia})})$ is a vector of $2(2\Delta + 1 + 1)$ inter- and intra-block cost parameters:

$$\boldsymbol{\theta}^{(\mathrm{ir})} = (\theta^{(\mathrm{ir})}_{-\Delta}, \ldots, \theta^{(\mathrm{ir})}_{\Delta}, \theta^{(\mathrm{ir})}_{\bullet}), \tag{4.3.8}$$
$$\boldsymbol{\theta}^{(\mathrm{ia})} = (\theta^{(\mathrm{ia})}_{-\Delta}, \ldots, \theta^{(\mathrm{ia})}_{\Delta}, \theta^{(\mathrm{ia})}_{\bullet}). \tag{4.3.9}$$

[8] CC-PEV stands for Cartesian Calibrated PEV features, see Section 4.2.5 for more details on Cartesian calibration.
[9] Different cropping in calibration caught the FCM that was built to approximately preserve PEV features, see Figure 4.3.1.

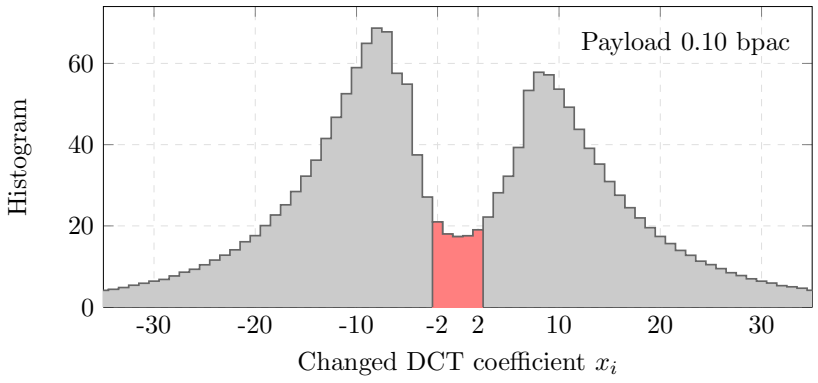


Figure 4.3.3: Histogram of changes to DCT coefficient values introduced by the MOD stegosystem with $\boldsymbol{\theta}^{(\mathrm{ia})} = 0$ at payload 0.10 bpac. The chart displays the average counts over 1,000 randomly selected images from the CAMERA database. The highlighted portion of the histogram around zero corresponds to the area covered by inter-block co-occurrences that are part of the CC-PEV model.

We adopt the convention $\theta_z^{(\mathrm{ia})} = \theta_\bullet^{(\mathrm{ia})}$, $\theta_z^{(\mathrm{ir})} = \theta_\bullet^{(\mathrm{ir})}$ whenever $|X_{ij} - z| > \Delta$. The subscript of each parameter corresponds to the difference between X_{ij} and its immediate neighbor, and the value of Δ controls the size of the parameter space and the complexity of the distortion function.

The authors optimized the parameters (4.3.8) and (4.3.9) for $\Delta = 6$ for the 548-dimensional CC-PEV cover model and stego images embedded with payload 0.5 bpac (bits per non-zero AC DCT coefficient). Two versions of the MOD algorithm were introduced – one in which both intra- and inter-block costs were optimized and the version in which only the inter-block parameters were optimized while $\boldsymbol{\theta}^{(\mathrm{ia})} \equiv (0, \ldots, 0)$. The latter one exhibited better security when tested with the CDF feature set.

The CC-PEV cover model considers various dependencies among DCT coefficients by forming co-occurrence matrices constrained to a rather limited range of $\{-2, \ldots, 2\}$ for inter-block neighbors and $\{-4, \ldots, 4\}$ for intra-block neighbors. A distortion function with parameters optimized w.r.t. this rather abruptly terminated model will likely underestimate the importance of dependencies among DCT coefficients outside of the range. Indeed, the MOD algorithm with $\boldsymbol{\theta}^{(\mathrm{ia})} = 0$ makes $\sim 95\%$ of all embedding changes to coefficients with absolute value greater than 2 (see Figure 4.3.3). Such changes are unlikely to be detected by the small-range co-occurrences in the CC-PEV model.

We confirm that this overtraining manifests in practice by first attacking the MOD algorithm with enlarged inter-block co-occurrences (IBCs). Formally, the feature vector $\mathbf{C}_{\mathrm{IBC}}(\mathbf{X})$ is a sum of two two-dimensional co-occurrence matrices:

$$\mathbf{C}_{\mathrm{IBC}}(\mathbf{X}) = \mathbf{C}^{\rightarrow}(\tilde{\mathbf{X}}) + \mathbf{C}^{\downarrow}(\tilde{\mathbf{X}}), \tag{4.3.10}$$

where $\tilde{\mathbf{X}} = \mathrm{trunc}_T(\mathbf{X})$, and the matrices $\mathbf{C}^k(\mathbf{X}) = (C^k_{\mathbf{d}}(\mathbf{X}))$, $k \in \{\rightarrow, \downarrow\}$, $\mathbf{d} \in \{-T, \ldots, T\}^2$, are defined element-wise as

$$C^{\rightarrow}_{\mathbf{d}}(\mathbf{X}) = \frac{1}{Z} \left| \{(i,j) | X_{i,j} = d_1 \wedge X_{i,j+8} = d_2\} \right|, \tag{4.3.11}$$

$$C^{\downarrow}_{\mathbf{d}}(\mathbf{X}) = \frac{1}{Z} \left| \{(i,j) | X_{i,j} = d_1 \wedge X_{i+8,j} = d_2\} \right|, \tag{4.3.12}$$

where Z is the normalization constant. With threshold T, the dimensionality of $\mathbf{C}_{\mathrm{IBC}}$ is $(2T+1)^2$. Figure 4.3.4 shows the results of detecting the MOD stegosystem (the version with $\boldsymbol{\theta}^{(\mathrm{ia})} = 0$) at a fixed payload 0.10 bpac using the IBC features, $\mathbf{C}_{\mathrm{IBC}}$, on the CAMERA database and a SVM classifier with the Gaussian kernel. For comparison, the detection error, P_{E}, for CC-PEV features

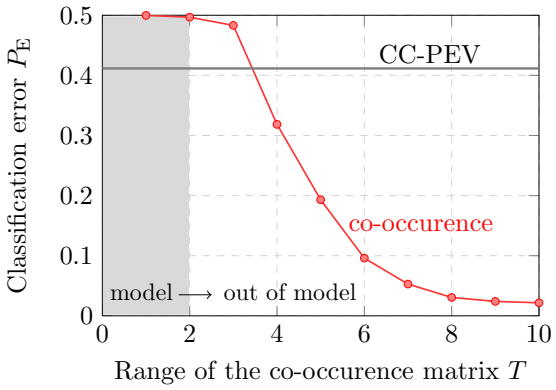


Figure 4.3.4: Detection error when steganalyzing the MOD stegosystem with $\boldsymbol{\theta}^{(\mathrm{ia})} = 0$ at payload 0.10 bpac using the IBC features (4.3.10) as a function of the threshold T.

is depicted with a horizontal line. Note that, according to our expectations, as soon as the IBC features get out of the CC-PEV model ($T = 2$), the detection error starts rapidly decreasing and reaches $P_E \approx 2\%$ with $T = 10$.

The accuracy of the attack can be further improved by also extending the intra-block part of the CC-PEV feature vector formed as a sum of four 9×9 conditional probability matrices modeling the differences between absolute values of neighboring DCT coefficients as a Markov process and thresholded with $T = 4$ (see [123, 107] for details). We enlarge this statistical descriptor by increasing the threshold to $T = 10$, obtaining thus a 441-dimensional feature vector, which we will refer to as the EM (Extended Markov) vector.

To complete the picture, we now steganalyze both versions of the MOD algorithm across a wider range of payloads using the union of IBC and EM features, both with $T = 10$ (model total dimensionality is $2 \times 441 = 882$). The results, shown in Figure 4.3.5, clearly support our argument that the MOD algorithm has been overtrained to an incomplete cover model. Extending the model decreases the security of the MOD algorithm to the extent that it is no longer more secure than the nsF5 algorithm (for payloads less then 0.2 bpac).

Note that the security of the inter-block-only optimized version of MOD ($\boldsymbol{\theta}^{(\mathrm{ia})} = 0$) is now much lower when compared to the case when both inter- and intra-block weights are optimized ($\boldsymbol{\theta}^{(\mathrm{ia})} \neq 0$). This should intuitively be the case as considering both types of dependencies leads to a more accurate (and complete) cover model that should be less prone to overtraining. We conjecture that the security of the MOD algorithm can likely be markedly improved by optimizing the costs w.r.t. an enlarged CC-PEV model.

Also notice that the security of nsF5 was not compromised by attacking it with IBC+EM features. In fact, the CC-PEV features are more successful in attacking nsF5 because the IBC+EM model lacks the diversity of the CC-PEV model and also because most changes made by nsF5 are made to DCT coefficients already covered by the small-range co-occurrences in the CC-PEV model.

4.4 Summary

This chapter equipped us with a solid understanding of feature space design foundations. Let us briefly review the most important lessons we have learned. A successful feature space should consist of many simpler submodels, each of them addressing a different aspect of the relationship between cover images and stego images. In Section 4.1, we discussed several different strategies for forming these submodels – they can be targeted to common embedding operations, they can be formed by

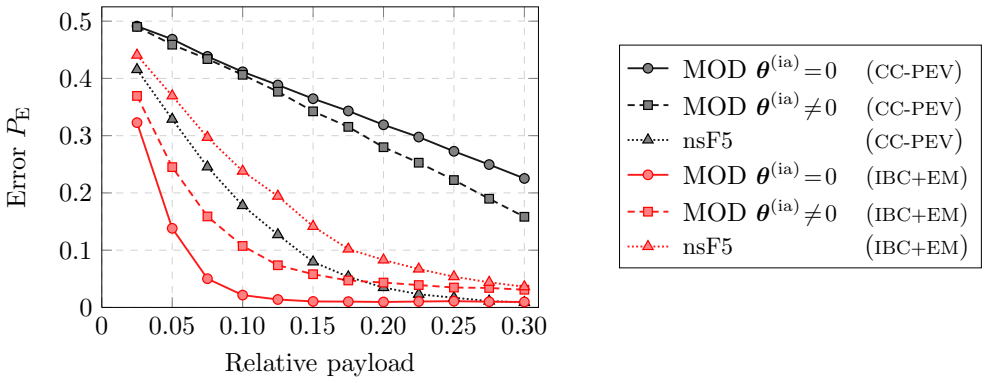


Figure 4.3.5: Detection error P_E for the MOD algorithm as proposed in [35] and nsF5 when attacked with CC-PEV features and the union of IBC and EM features.

adopting different simplified image models, or by modeling noise residuals rather than the cover coefficients directly. The power of these qualitatively different approaches consists in combining them together and using their union as a final feature space. The individual elements of differently constructed submodels will likely be dependent and will serve each other as references which could be interpreted as an instance of Cartesian calibration discussed in Section 4.2. In fact, we have shown that it may be beneficial to include a certain number of submodels, often with a very poor standalone distinguishing power, whose purpose is *solely* to serve as a reference value to other submodels – Cartesian calibration is an important element of modern feature space design and we consider it as a powerful model enrichment technique.

Section 4.3 provided us with a valuable insight into potential pitfalls we need to consider when building feature spaces. We need to pay attention to the boundaries of our models and try to avoid creating security holes that could be utilized by a steganalyst. Feature spaces should be as diverse as possible and as complete as possible – they should statistically cover all the image coefficients and the most important dependencies among them. This may lead to very high-dimensional feature spaces, however, we believe that this should not be an obstacle to the feature space design, which is why we proposed ensemble classifiers that can handle high dimensions very well (see Chapter 3).

In the next two chapters, we utilize the acquired knowledge and construct very diverse and rich feature spaces for steganalysis in the spatial domain and in the DCT domain.

Chapter 5

Spatial domain rich model (SRM)

In this chapter, we construct a rich image model for spatial domain steganalysis that consists of a large number of diverse submodels. It is built in the spatial domain because the best detection is usually achieved by building the model directly in the domain where the embedding changes are localized and thus most pronounced. The submodels consider various types of relationships among neighboring samples of noise residuals obtained by linear and non-linear filters with compact supports. Our design, inspired by the recent methods developed for attacking HUGO [50, 49, 54], brings this philosophy to the next level by creating submodels in a more systematic and exhaustive manner. We describe the model in all details in Section 5.1.

In Section 5.2, we present a series of investigative experiments whose purpose is to study the importance of individual submodels for steganalysis of different stegosystems. Furthermore, we combine the feature space construction with a classification feedback, and by introducing a few heuristic model-assembling strategies we identify a combination of submodels that achieves a good trade-off between model dimensionality and detection accuracy. This can be viewed as a step towards automatizing steganalysis to facilitate fast development of accurate detectors for new steganographic schemes. Since the experiments require fast machine learning, we conveniently use the ensemble classifier described in Chapter 3.

In Section 5.3, the steganalysis performance of the full framework is experimentally evaluated on three qualitatively different spatial-domain steganographic algorithms. The results show that the proposed framework outperforms other existing approaches. We also discuss the possibility of using a Gaussian SVM as a final classifier, once a good low-dimensional subspace of the rich model is identified.

The chapter is summarized in Section 5.5 where we also elaborate on how the proposed strategy affects future development of steganography and discuss potential applications of rich models outside the field of steganalysis.

5.1 Building the rich model

We construct the rich model by considering the noise component (noise residual) of images rather than their content. The advantage of modeling the residual instead of the pixel values is that the image content is largely suppressed, narrowing thus a dynamic range and allowing for a more compact and robust statistical description. This strategy is consistent with the insight gained in Chapter 4.1.3, and has been adopted by many steganalysts in the past, e.g., [6, 5, 31, 52, 104, 137].

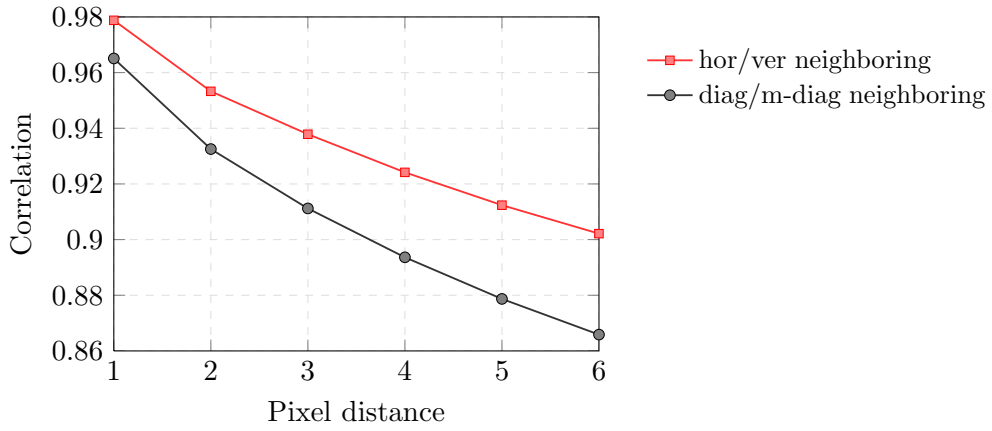


Figure 5.1.1: Correlation between pixels based on their distance. The distance of diagonally neighboring pixels is in the multiples of the diagonal of two neighboring pixels. The results were averaged over 100 randomly selected images of BOSSbase ver. 0.92.

5.1.1 Submodels

The individual submodels of the proposed rich model are formed by joint distributions of neighboring samples from quantized image noise residuals obtained using linear and non-linear high-pass filters with compact supports. Formally, for an image $\mathbf{X} = (X_{ij}) \in \mathbb{R}^{n_1 \times n_2}$, a noise residual, $\mathbf{R} = (R_{ij}) \in \mathbb{R}^{n_1 \times n_2}$, is computed using a high-pass filter of the following form:

$$R_{ij} = \hat{X}_{ij}(\mathcal{N}_{ij}) - cX_{ij}, \tag{5.1.1}$$

where $c \in \mathbb{N}$ is the residual order, $\mathcal{N}_{ij}$ is a local neighborhood of pixel X_{ij}, $X_{ij} \notin \mathcal{N}_{ij}$, and $\hat{X}_{ij}(\cdot)$ is a *predictor* of cX_{ij} defined on $\mathcal{N}_{ij}$. The set $\{X_{ij} \cup \mathcal{N}_{ij}\}$ is called the support of the residual.

Each submodel is formed from a quantized and truncated version of the residual:

$$R_{ij} \leftarrow \mathrm{trunc}_T \left(\mathrm{round} \left(\frac{R_{ij}}{q} \right) \right), \tag{5.1.2}$$

where $q > 0$ is a quantization step. The purpose of truncation, properly defined in Chapter 1.5 by (1.5.1), is to curb the residual's dynamic range to allow their description using co-occurrence matrices with a relatively small T. The quantization makes the residual more sensitive to embedding changes at spatial discontinuities in the image (at edges and textures).

The construction of each submodel continues with computing one or more co-occurrence matrices of neighboring samples from the truncated and quantized residuals (5.1.2). Forming models in this manner is well-established in the steganalysis literature. The key question is how to choose the model parameters – the threshold T, the co-occurrence order, and the spatial positions of the neighboring residual samples. To this end, we analyzed our cover source, which is the BOSSbase database ver. 0.92 (described in Appendix B), and computed the average correlation between neighboring pixels in the horizontal/vertical and diagonal/minor diagonal directions (see Figure 5.1.1). The correlations fall off gradually with increasing distance between the pixels and they do so faster for diagonally-neighboring pixels. Thus, we form co-occurrences of pixels only along the horizontal and vertical directions and avoid using groups with diagonally-neighboring pixels.[1] We chose four-dimensional co-occurrences because co-occurrences of larger dimensions had numerous underpopulated bins, which

[1] A few sample tests confirmed that co-occurrences formed from groups in which pixels do not lie on a straight line have a substantially weaker detection performance across various stego methods and payloads.

compromised their statistical significance. For a fixed dimension, better results are generally obtained by using a lower value of T and including other types of residuals to increase the model diversity. To compensate for loss of information due to truncating all residual values larger than T, for each residual type we consider several submodels with different values of q, allowing thus our model to "see" dependencies among residual samples whose values lie beyond the threshold.

In summary, our submodels will be constructed from horizontal and vertical co-occurrences of four consecutive residual samples processed using (5.1.2) with $T = 2$. Formally, each co-occurrence matrix $\mathbf{C}$ is a four-dimensional array indexed with $\mathbf{d} = (d_1, d_2, d_3, d_4) \in \mathcal{T}_4 \triangleq \{-T, \dots, T\}^4$, which gives the array $(2T+1)^4 = 625$ elements. The $\mathbf{d}$th element of the horizontal co-occurrence for residual $\mathbf{R} = (R_{ij})$ is formally defined as the (normalized) number of groups of four neighboring residual samples with values equal to d_1, d_2, d_3, d_4:

$$C^{\rightarrow}_{\mathbf{d}}(\mathbf{R}) = \frac{1}{Z}\Big|\{(R_{ij}, R_{i,j+1}, R_{i,j+2}, R_{i,j+3}) \mid R_{i,j+k-1} = d_k, k = 1, \dots, 4\}\Big|, \tag{5.1.3}$$

where Z is the normalization factor ensuring that $\sum_{\mathbf{d}\in\mathcal{T}_4} C^{\rightarrow}_{\mathbf{d}} = 1$. The vertical co-occurrence is defined analogically.

Having fixed T and the co-occurrence order, determining the rest of the rich model involves selecting the local predictors $\hat{X}_{ij}$ for the residuals and the quantization step(s) q, all explained in the following subsections.

5.1.2 Description of all residuals

All our residuals are graphically shown in Figure 5.1.2. They are built as locally-supported linear filters whose outputs are possibly combined using minimum and maximum operators to increase their diversity. For better insight, think of each filter in terms of its predictor. For example, in the first-order residual $R_{ij} = X_{i,j+1} - X_{ij}$ the central pixel X_{ij} is predicted as its immediate neighbor, $\hat{X}_{ij} = X_{i,j+1}$, while the predictor in the second-order residual $R_{ij} = X_{i,j-1} + X_{i,j+1} - 2X_{ij}$ assumes that the image is locally linear in the horizontal direction, $2\hat{X}_{ij} = (X_{i,j+1} + X_{i,j-1})$. Higher-order differences as well as differences involving a larger neighborhood correspond to more complicated assumptions made by the predictor, such as locally-quadratic behavior or linearity in both dimensions.

The central pixel X_{ij} at which the residual (5.1.1) is evaluated is always marked with a black dot in Figure 5.1.2, and accompanied with an integer – the value c from (5.1.1). If the chart contains only one type of symbol (besides the black dot), we say that the residual is of type 'spam' (1a, 2a, 3a, S3a, E3a, S5a, E5a) by its similarity to the SPAM feature vector [104].

If there are two or more different symbols other than the black dot, we call it type 'minmax'. In type 'spam', the residual is computed as a linear high-pass filter of neighboring pixels with the corresponding coefficients. For example, 2a stands for the second-order $R_{ij} = X_{i,j-1} + X_{i,j+1} - 2X_{ij}$ and 1a for the first-order $R_{ij} = X_{i,j+1} - X_{ij}$ residuals. In contrast, 'minmax' residuals use two or more linear filters (each filter corresponding to one symbol type in Figure 5.1.2), and the final residual is obtained by taking the minimum (or maximum) of the filters' outputs. Thus, there will be two minmax residuals – one for the operation of 'min' and one for 'max'. For example, 2b is obtained as $R_{ij} = \min\{X_{i,j-1} + X_{i,j+1} - 2X_{ij}, X_{i-1,j} + X_{i+1,j} - 2X_{ij}\}$ while 1g is $R_{ij} = \min\{X_{i-1,j-1} - X_{ij}, X_{i-1,j} - X_{ij}, X_{i-1,j+1} - X_{ij}, X_{i,j+1} - X_{ij}\}$, etc. The 'min' and 'max' operators introduce non-linearity into the residuals and desirably increase the model diversity. Both operations also make the distribution of the residual samples non-symmetrical, thickening one tail of the distribution of R_{ij} and thinning out the other.

The number of filters, f, is the first digit attached to the end of the residual name. The third-order residuals are computed just like the first-order residuals by replacing, e.g., $X_{i,j+1} - X_{ij}$ with $-X_{i,j+2} + 3X_{i,j+1} - 3X_{ij} + X_{i,j-1}$. The differences along other directions are obtained analogically.

Figure 5.1.2: Definitions of all residuals. The residuals 3a – 3h are defined similarly to the first-order residuals, while E5a – E5d are similar to E3a – E3d defined using the corresponding part of the 5×5 kernel displayed in S5a. See the text for more details.

Residual classes

As Figure 5.1.2 shows, the residuals are divided into six classes depending on the central pixel predictor they are built from. The classes are given the following descriptive names: 1st, 2nd, 3rd, SQUARE, EDGE3x3, and EDGE5x5. The predictors in class '1st' estimate the pixel as the value of its neighbor, while those from class '2nd' ('3rd') incorporate a locally linear (quadratic) model. Such predictors are more accurate in regions with a strong gradient/curvature (e.g., around edges and in complex textures). The class 'SQUARE' makes use of more pixels for the prediction. The 3×3 square kernel S3a has been used in steganalysis before [68] and it also coincides with the best (in the least-square sense) shift-invariant linear pixel predictor on the 3×3 neighborhood for cover images from BOSSbase. The class 'EDGE3x3' predictors, derived from this kernel, was included to provide better estimates at spatial discontinuities (edges). The larger 5×5 predictor in S5a was obtained as a result of optimizing the coefficients of a circularly-symmetrical 5×5 kernel using the Nelder–Mead algorithm to minimize the detection error for the embedding algorithm HUGO [80]. While this (only) predictor was inspired by a specific embedding algorithm, it works very well against other algorithms we tested in this chapter. The 'EDGE5x5' residuals E5a–E5d (not shown in Figure 5.1.2) are built from S5a in an analogical manner as E3a–E3d are built from S3a.

Residual symmetries

Each residual exhibits symmetries that will later allow us to reduce the number of submodels and make them better populated. If the residual does not change after computing it from the image rotated by 90 degrees, we say that it is non-directional, otherwise it is directional. For instance, 1a, 1b, 2a, 2e, E3c are directional while 1e, 2b, 2c, S3a, E3d are non-directional. Two co-occurrence matrices (5.1.3) are computed for each residual – one for the horizontal and one for the vertical scan. We call a residual hv-symmetrical if its horizontal and vertical co-occurrences can be added to form a single matrix (submodel) based on the argument that the statistics of natural images do not change after rotating the image by 90 degrees. Obviously, all non-directional residuals are hv-symmetrical, but many directional residuals are hv-symmetrical as well (e.g, 1c, 1h, 2e, E3b, E3d). In contrast, 1a, 1g, 2a, 2d, E3c are not hv-symmetrical. In general, an hv-symmetrical residual will thus produce a single co-occurrence matrix (sum of both horizontal and vertical matrices), while hv-nonsymmetrical ones will produce two matrices – one for the horizontal and one for the vertical direction. We include this fact into the residual name by appending either 'h' or 'v' to the end. No symbol is appended to hv-symmetrical residuals.

We also define a symmetry index σ for each residual as the number of different residuals that can be obtained by rotating and possibly mirroring the image prior to computing it. To give an example, 2c, 1b, 1c, and 1g have symmetry indices equal to 1, 2, 4, and 8, respectively. The symmetry index is part of the residual name and it always follows the number of filters, f.

To make the co-occurrence bins more populated, and thus increase their statistical robustness, and to lower their dimensionality, for hv-nonsymmetrical residuals we add all σ co-occurrences. For hv-symmetrical residuals, since we add both the horizontal and vertical co-occurrences, we end up adding 2σ matrices. For example, 1f has symmetry index 4 and because it is hv-symmetrical we can form one horizontal and one vertical co-occurrence for each of the four rotations of the filter, adding together 8 matrices. As another example, 1g has symmetry index 8 and is hv-nonsymmetrical, which means we end up adding 8 matrices.

Syntax

The syntax of names used in Figure 5.1.2 follows this convention:

$$name = \{type\}\{f\}\{\sigma\}\{scan\}, \tag{5.1.4}$$

where $type \in \{\text{spam}, \text{minmax}\}$, f is the number of filters, σ is the symmetry index, and the last symbol $scan \in \{\emptyset, h, v\}$ may be missing (for hv-symmetrical residuals) or it is either h or v, depending on the co-occurrence scan that should be used with the residual.

In summary, the class '1st' contains 22 different co-occurrence matrices – two for 1a, 1c, 1e, 1f, 1h, and four for 1b, 1d, 1g. The same number is obtained for class '3rd', while '2nd' contains 12 matrices – two for 2a, 2b, 2c, 2e, and four for 2d. There are two matrices in 'SQUARE', S3a, S5a, and ten in 'EDGE3x3' and in 'EDGE5x5' (two for E3a, E3b, and E3d, and four for E3c), giving the total of $22 + 12 + 22 + 2 + 10 + 10 = 78$ matrices, each with 625 elements. These matrices are used to form the final submodels by *co-occurrence* symmetrization explained next.

5.1.3 Co-occurrence symmetrization

The individual submodels of the rich image model will be obtained from the 78 co-occurrence matrices computed above by leveraging symmetries of natural images. The symmetries are in fact quite important as they allow us to increase the statistical robustness of the model while decreasing its dimensionality, making it thus more compact and obtaining a better performance-to-dimensionality ratio. We use the sign-symmetry[2] as well as the directional symmetry of images. The symmetrization depends on the residual type. All 'spam' residuals are symmetrized sequentially by applying the following two rules for all $\mathbf{d} = (d_1, d_2, d_3, d_4) \in \mathcal{T}_4$:

$$\bar{C}_{\mathbf{d}} \leftarrow C_{\mathbf{d}} + C_{-\mathbf{d}}, \tag{5.1.5}$$

$$\bar{\bar{C}}_{\mathbf{d}} \leftarrow \bar{C}_{\mathbf{d}} + \bar{C}_{\overleftarrow{\mathbf{d}}}, \tag{5.1.6}$$

where $\overleftarrow{\mathbf{d}} = (d_4, d_3, d_2, d_1)$ and $-\mathbf{d} = (-d_1, -d_2, -d_3, -d_4)$. After eliminating duplicates from $\bar{\bar{\mathbf{C}}} = (\bar{\bar{C}}_{\mathbf{d}})$ (which had originally 625 elements), only 169 unique elements remain.

The 'minmax' residuals of natural images also possess the directional symmetry but not the sign symmetry. On the other hand, since $\min(\mathcal{X}) = -\max(-\mathcal{X})$ for any finite set $\mathcal{X} \subset \mathbb{R}$, we use the following two rules for their symmetrization:

$$\bar{C}_{\mathbf{d}} \leftarrow C^{(\min)}_{\mathbf{d}} + C^{(\max)}_{-\mathbf{d}} \tag{5.1.7}$$

$$\bar{\bar{C}}_{\mathbf{d}} \leftarrow \bar{C}_{\mathbf{d}} + \bar{C}_{\overleftarrow{\mathbf{d}}}, \tag{5.1.8}$$

where $\mathbf{C}^{(\min)}$ and $\mathbf{C}^{(\max)}$ are the 'min' and 'max' co-occurrence matrices computed from the same residual. The dimensionality is thus reduced from 2×625 to 325.

After symmetrization, the total number of submodels decreases from 78 to only 45 as the symmetrization reduces two co-occurrences, one for 'min' and one for 'max', into a single matrix. The number of co-occurrences for the 'spam' type stays the same (only their dimensionality changes). For example, for the class '1st', we will have 12 submodels – one symmetrized spam14h and one spam14v, one minmax22h, one minmax22v, one minmax24, minmax34h, minmax34v, minmax41, minmax34, minmax48h, minmax48v, and one minmax54. There will be 12 submodels from '3rd', seven from '2nd', two from 'SQUARE', and six from each edge class. In total, there are 12 submodels of dimension 169 from 12 'spam' type residuals and 33 of dimension 325 from type minmax. Thus, when all submodels are put together, their combined dimensionality is $12 \times 169 + 33 \times 325 = 12,753$.

We remark that it is possible that the symmetrization might prevent us from detecting steganographic methods that disturb the above symmetries (think of symmetrizing the histogram for Jsteg [129]). Such embedding methods are, however, fundamentally flawed and it is unlikely that they would leave other dependencies captured by the rich model undisturbed. Furthermore, one can build accurate quantitative targeted attacks leveraging the symmetry violations.

[2] Sign-symmetry means that taking a negative of an image does not change its statistical properties.

5.1.4 Quantization

Finally, we specify how to select the quantization step q. As mentioned already at the end of Section 5.1.2, it is beneficial to include several versions of submodels with different values of q because residuals obtained with a larger q are better able to detect embedding changes in textured areas and around edges. Based on sample experiments with different algorithms and submodels, we determined that the best performance of each submodel is always achieved when $q \in [c, 2c]$, where c is the residual order. Thus, we included in the rich model all submodels with residuals quantized with q:

$$q \in \begin{cases} \{c, 1.5c, 2c\} & \text{for } c > 1 \\ \{1, 2\} & \text{for } c = 1. \end{cases} \tag{5.1.9}$$

The case with $c = 1$ in (5.1.9) is different from the rest because quantizing a residual with $c = 1$ and $q = 1.5$ with $T = 2$ leads to exactly the same result as when quantizing with $q = 2$. Thus, each submodel will be built in two versions for residuals in class '1st' (as only these have $c = 1$) and in three versions for the remaining residuals.

The union of all submodels (co-occurrence matrices), including their differently quantized versions, has dimension $2 \times (2 \times 169 + 10 \times 325) + 3 \times (10 \times 169 + 23 \times 325) = 34,671$.

5.1.5 Discussion

The residuals shown in Figure 5.1.2 were selected using the principle of simplicity and are by no means to be meant as the ultimate result as there certainly exist numerous other possibilities. We view the model building as an open-ended process because, quite likely, there exist other predictors that will further improve the detection after adding them to the proposed model. Having said this, we observed a "saturation" of performance in the sense that further enrichment of the model with other types of predictors lead to an insignificant improvement in detection accuracy for all tested algorithms (see Experiment 2 in Section 5.2.2).

We also note that submodels obtained from residuals computed using denoising filters almost always lead to poor steganalysis results because denoising filters typically put substantial weight to the central pixel being denoised, which leads to a biased predictor $\hat{X}_{ij}$, and, when one computes the residual using (5.1.1), the stego signal becomes undesirably suppressed.

5.2 Investigative experiments

After the construction of the 34,671-dimensional Spatial domain Rich Model (SRM), we conduct a few investigative steganalysis experiments. All experiments are carried out on three different steganographic algorithms with contrasting embedding mechanisms. The first is the simple non-adaptive ± 1 embedding (also called LSB matching) implemented with optimally coded ternary matrix embedding (see Appendix A.12). The second algorithm is HUGO [105] – we used the embedding simulator available from the BOSS website[3] with $\sigma = 1$ and $\gamma = 1$ for the parameters of the distortion function, and the switch –T 255, which means that the distortion function was computed with threshold $T = 255$ instead of the default value $T = 90$ used in the BOSS challenge [9]. We did it to remove a weakness of HUGO with $T = 90$ that was revealed and discussed in Chapter 4.3.3 of this dissertation. The third, Edge-Adaptive (EA) algorithm, due to Luo *et al.* [94], confines the embedding changes to pixel pairs whose difference in absolute value is as large as possible (e.g., around edges). This algorithm was included intentionally as an example of a stegosystem that, according to the best of our knowledge, has not yet been successfully attacked. Both HUGO and the EA algorithm, also briefly described in the Appendix, place the embedding changes to those parts

[3] http://www.agents.cz/boss

Class	List of submodels (acronyms)	q	# of models	Total dimension
1st	22h 22v 24 34 34h 34v 41 48h 48v 54 14hv	1, 2	$2 \times 11 = 22$	$2 \times (10 \times 325 + 338) = 7,176$
2nd	21 24h 24v 32 41 12hv	1, 1.5, 2	$3 \times 6 = 18$	$3 \times (5 \times 325 + 338) = 5,889$
3rd	22h 22v 24 34 34h 34v 41 48h 48v 54 14hv	1, 1.5, 2	$3 \times 11 = 33$	$3 \times (10 \times 325 + 338) = 10,764$
EDGE3x3	22h 22v 24 41 14hv	1, 1.5, 2	$3 \times 5 = 15$	$3 \times (4 \times 325 + 338) = 4,914$
EDGE5x5	22h 22v 24 41 14hv	1, 1.5, 2	$3 \times 5 = 15$	$3 \times (4 \times 325 + 338) = 4,914$
SQUARE	11	1, 1.5, 2	$3 \times 1 = 3$	$3 \times 338 = 1,014$
TOTAL			106	34,671

Table 5.1: The list of all 106 submodels of the proposed rich cover model. The acronym of each submodel consists of the number of filters, f, the symmetry index, σ, and the scan direction. For submodels of type 'spam', since both scan directions were merged, we use the scan string 'hv'. The quantization step q is shown in multiples of c (see Equation (5.1.9)).

of the image that are hard to model and are thus expected to be more secure than the non-adaptive ± 1 embedding.

The image source used for all experiments is the BOSSbase ver. 0.92. The reason for constraining our investigation to a single cover source was our desire to focus on the methodology rather than benchmarking steganography in different cover sources.

5.2.1 Experiment 1

The goal of Experiment 1 is to obtain insight about the detection performance for each submodel, stego algorithm, and quantization step. For this purpose, we rank the submodels according to their individual OOB error estimate (3.2.1) calculated from the training set. We intentionally exclude the feedback from the testing set for the performance evaluation because the resulting submodel ranking will be further utilized in Experiment 2 and for the final testing of the whole framework in Section 5.3, and therefore no testing feedback is allowed. Note that the OOB error estimate is, indeed, ideally suited for evaluating the detection performance of the submodel on unseen data as it is an unbiased estimate of the testing error and it is a convenient "by product" of the ensemble training. Therefore we do not need to reserve a portion of the training set to assess the testing error as is commonly done in cross-validation.

The rich model constructed in Section 5.1 consists of submodels of different dimensions – the dimensionality of submodels that originated from the 'spam' type residual is 169 and 'minmax' residuals have dimensionality 325. To make the comparison of submodels fair, we merge the vertical and horizontal 'spam' submodels into a single $2 \times 169 = 338$-dimensional submodel (merging spam14h with spam14v from classes '1st', '3rd', 'EDGE3x3', and 'EDGE5x5', and also spam12h with spam12v from class '2nd'). Additionally, we merge the two spam11 submodels from class 'SQUARE' into one 338-dimensional submodel. Now, all submodels have approximately the same dimension (338 or 325) and can thus be fairly compared. After this merger, we have 11 submodels in class '1st' and '3rd', six in '2nd', one in 'SQUARE', and five in each edge class. That leads to the total of 39 submodels, or 106 if counting quantized versions as different. See Table 5.1 for a comprehensive list of all 106 submodels. The table also contains a short acronym of each submodel that will be used in the graph of Figure 5.2.1.

The Matlab extractor of all 106 submodels (34,671 features in total) is available at `http://dde.binghamton.edu/download/feature_extractors`.

We compute the OOB estimates for each submodel, including its differently quantized versions, for each stego method and for one small and one large payload (0.1 and 0.4 bpp). Since the ensemble classifier is built using random structures (randomness enters the selection of subspaces for base

Algorithm	±1 embedding		HUGO		EA	
Payload	0.10	0.40	0.10	0.40	0.10	0.40
avg. MAD	0.649	0.679	0.656	0.526	0.601	0.484
max. MAD	1.640	1.500	2.840	1.220	1.200	0.940

Table 5.2: Mean Absolute Deviation (MAD) of OOB estimates ($\times 10^{-3}$) over five independent ensemble realizations on the same training/testing split. The table reports the average and maximal values over all 106 submodels listed in Table 5.1 for all three tested stego algorithms and two payloads.

learners and the bootstrap sample formation), we repeated each run five times, each of them on the same split of BOSSbase into 8,074 training and 1000 testing images, and report the average values of OOB estimates. Table 5.2 shows that the variations are in general rather negligible. Note that they can be made arbitrarily small by the user by increasing the number of the base learners L.

The results are summarized in Figure 5.2.1 showing the OOB error estimates for all 39 submodels and for all values of q. The vertical dashed lines separate the 'spam' submodels from submodels of type 'minmax'. The dots were connected by a line to enable a faster visual interpretation of the results.

Evaluating individual algorithms

By comparing the patterns for a fixed algorithm, we see that there is a great deal of similarity between the performance of submodels across payloads even though the actual rankings may be different. Remarkably, ±1 embedding with small payload shows by far the largest sensitivity to the quantization factor than any other combination of algorithms and payloads. This effect is caused by the non-adaptive character of ±1 embedding. For small payloads, the amount of changes in edges and textures is so small that detection essentially relies on smooth parts where the finest quantization discerns the embedding far better in comparison to other quantizations. On the contrary, both adaptive algorithms are much less sensitive to the quantization step because small payloads are more concentrated in textures and edges.

Notice that, for HUGO, submodels built from first-order differences have worse performance than submodels obtained from third-order differences, which is due to the fact that HUGO approximately preserves statistics among first-order differences; the higher-order differences thus reach "beyond" the model. Also, features of type 'spam' seem to be consistently better than 'minmax' for this algorithm.

The OOB estimates for the EA algorithm exhibit a remarkable similarity for both payloads. This property can probably be attributed to the much more selective character of embedding. While HUGO makes embedding changes even in less textured areas albeit with smaller probability, the EA algorithm limits the embedding only to those pairs of adjacent pixels whose difference is above a certain threshold, eliminating a large portion of the image from the embedding process.

Universality of submodels

The best individual submodels for the larger payload and ±1 embedding, HUGO, and EA achieve OOB error estimates around 0.1, 0.21, and 0.12, indicating that HUGO is by far the best algorithm among the three. While there exist clear differences among the performance of each submodel across algorithms, it is worth noting that certain submodels rank the same w.r.t. each other for all three algorithms, both payloads, and all quantization steps. For example, 'minmax22h' is always worse than 'minmax22v' for class '1st' as well as '3rd'. In other words, it is better to form co-occurrences in

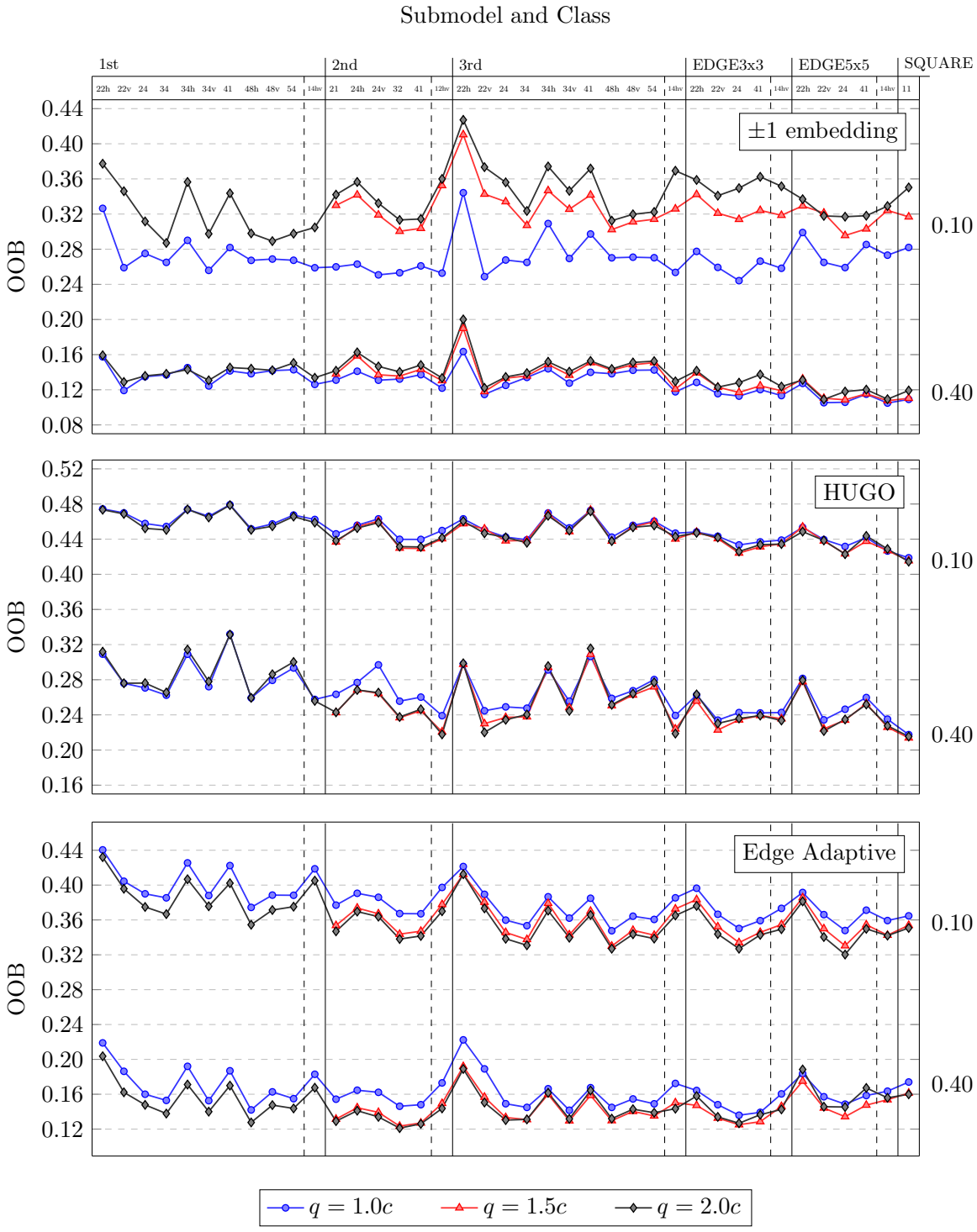


Figure 5.2.1: OOB error estimates (3.2.1) for all 106 submodels listed in Table 5.1 for three stego algorithms and two payloads. The values were averaged over five runs of the ensemble for a fixed split of the BOSSbase.

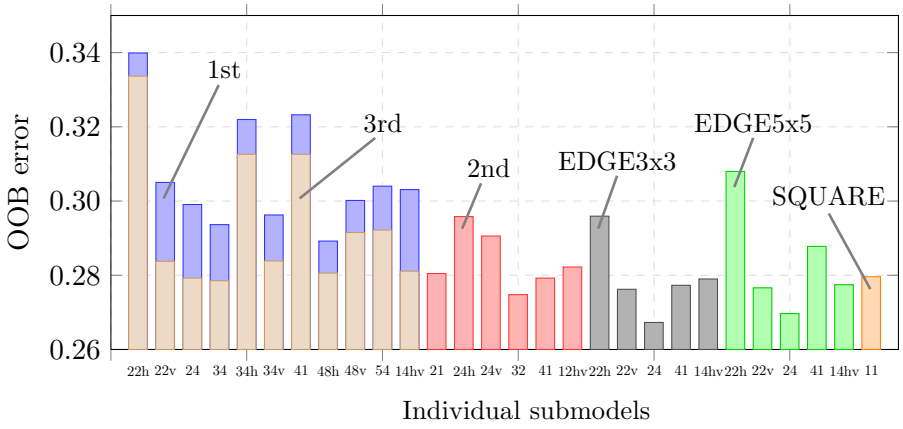


Figure 5.2.2: OOB error estimates averaged over all three stego methods and five payloads (0.05, 0.1, . . ., 0.4 bpp). Individual classes are shown in different colors.

the direction that is perpendicular to the direction in which the pixel differences are computed. This is most likely because the perpendicular scan prevents overlaps of filter supports and thus utilizes more information among neighboring pixels. The universality of submodels is further supported by the fact that pair-wise relationships between submodels are largely invariant to stego method and payload – for 40% of all pairs of submodels, the numerical relationship between their OOB errors does not depend on the algorithm or payload.

To further investigate the universality of submodels, in Figure 5.2.2 we plot for each submodel its OOB error estimate averaged over all three stego algorithms and five payloads, 0.05, 0.1, 0.2, 0.3, and 0.4 bpp. The best overall submodel is minmax24 in class 'EDGE3x3'. Note that 'h' versions of submodels built from residuals that are not hv-symmetrical are almost always worse than 'v' versions as most residuals are defined in Figure 5.1.2 in their horizontal orientation. This supports the rule that forming co-occurrence matrices in the direction perpendicular to the orientation of the kernel support generally leads to better detection as the co-occurrence bin utilizes more pixels. Figure 5.2.2 also nicely demonstrates that submodels built from first-order differences are in general worse than their equivalents constructed from third-order differences. Finally, observe that the best submodels are in general from hv-symmetrical non-directional residuals.

5.2.2 Experiment 2

In Experiment 2, we apply various feature selection strategies to find a low-dimensional subset of the rich model responsible for most of the detection accuracy. The feature selection is applied for a given steganographic algorithm and a sample of cover and stego images or, perhaps more accurately, for a given stegochannel (as defined in Chapter 2.1), which includes a specific choice of the message source (payload size).

We apply feature selection strategies to the *entire submodels* rather than to individual features as this allows us to interpret the results, relate the selected submodels to the steganographic algorithms, and provide interesting feedback to steganographers, which would not be possible otherwise.

Denoting with $\mathcal{M}_i^{(q)}$ the ith submodel, $i = 1, \ldots, 39$, quantized with q ($q \in \{1c, 2c\}$ for i in class '1st' and $q \in \{1c, 1.5c, 2c\}$ otherwise) and its OOB error estimate $E_i^{(q)}$, we explore the following simple feature selection strategies:

- ALL. Start with all 106 submodels, rank them by $E_i^{(q)}$, and build the feature space by adding the submodels by their rank, starting with the lowest $E_i^{(q)}$.

- BEST-q. First, compute $q_i = \arg\min_q E_i^{(q)}$ for each i and select all 39 submodels $\mathcal{M}_i^{(q_i)}$. Build the feature space by merging them according to their $E_i^{(q_i)}$, starting with the lowest. Heuristically, since two differently quantized versions of one submodel provide less diversity than two submodels built from different residuals, we expect this strategy to provide better performance-to-dimensionality ratio than ALL.
- BEST-q-CLASS. As in BEST-q, start with 39 submodels $\mathcal{M}_i^{(q_i)}$ and select them by $E_i^{(q_i)}$ but do it in repeated rounds, each round consisting of selecting the best submodel $\mathcal{M}_i^{(q_i)}$ from each class and eliminating that submodel from the class. The idea is to force diversity even stronger than in BEST-q as the first six submodels are guaranteed to be selected from six different classes, etc.
- Q1. Merge $\mathcal{M}_i^{(1c)}$, $i = 1, \ldots, 39$. We want to compare fixed quantization $q = 1c$ with the optimized quantization of the BEST-q (or BEST-q-CLASS) strategy after merging all 39 submodels.
- CLASS-q. Form the model by merging all submodels with the best quantization step q from a fixed class. The goal is to see how successful each residual *type* is in detecting a given algorithm.
- ITERATIVE-BEST-q. This strategy is the only one that considers mutual dependencies among submodels. The submodels are selected sequentially one by one based on how much they improve the detection w.r.t. the union of those already selected. We start with 39 submodels just like in strategies BEST-q and BEST-q-CLASS. The first submodel selected is the one with the lowest OOB error. Having selected $k \geq 1$ submodels, add the one among the $N_s - k$ remaining ones that leads to the biggest drop in the OOB estimate when all $k + 1$ submodels are used as a model.

From the machine learning point of view, the first three strategies, ALL, BEST-q, and BEST-q-CLASS, could be classified as filters [87]. They are based solely on the initial individual OOB ranking of every submodel (Experiment 1) and thus dependencies among the submodels are ignored. They differ in the amount of imposed diversity. Strategy ITERATIVE-BEST-q, on the other hand, continuously utilizes classification feedback of the ensemble and attempts to greedily minimize the OOB error at every iteration. This way, the mutual dependencies among individual submodels are taken into account. Strategy ITERATIVE-BEST-q is an example of a *wrapper* feature selection method [87] which uses a machine learning tool as a black-box and thus the results are classifier-dependent. Filters and wrappers are known as *forward* feature selection methods.

The CLASS-q strategy corresponds to merging all submodels with the best q from one chosen class, while the Q1 strategy corresponds to merging all 39 submodels with a fixed quantization $q = 1c$. The purpose of these two simple heuristic merging strategies is rather investigative.

Evaluation of individual submodel selection strategies

The efficiency of the proposed submodel-selection strategies is studied for a fixed payload of 0.4 bpp for all three algorithms on the training set for one fixed split of BOSSbase into 8074 training and 1000 testing images. Figure 5.2.3 shows the OOB error estimate as a function of model dimensionality for all assembly strategies. Diversity-boosting strategies (BEST-q and BEST-q-CLASS) clearly achieve better results than merging submodels based solely on their individual detection performance (ALL).

As expected, ITERATIVE-BEST-q outperforms all other strategies but its complexity limited us to merging only ten submodels. A little over 3000 features are in general sufficient to obtain detection accuracy within $0.5 - 1\%$ of the result when the entire 34,761-dimensional rich model is used. When all ten submodels are selected using the ITERATIVE-BEST-q strategy, the best "dependency-unaware" strategy, BEST-q-CLASS, needs roughly double the dimension for comparable performance. This seems to suggest that further and probably substantial improvement of

the performance-dimensionality trade-off is likely possible using more sophisticated feature-selection methods.

Overall, the lowest OOB error estimate is indeed obtained when all 106 (dimension 34,671) submodels are used. The gain between using 39 submodels (dimensionality 12,753) and all 106 quantized submodels is however rather negligible, indicating a saturation of performance.

Models assembled from a specific class (CLASS-q) also provide interesting insight. We obtain another confirmation that third-order residuals have better detection accuracy than first-order residuals across all stego algorithms. Remarkably, despite its lower dimension, the model assembled from class '2nd' for HUGO is better than class '1st'. This is not true for the other two algorithms and is most likely due to the fact that HUGO preserves complex statistics computed from first-order differences among neighboring pixels. Curiously, while 'EDGE5x5' is better than 'EDGE3x3' for ± 1 embedding and HUGO, the opposite is true for EA. The 'EDGE5x5' class appears to be particularly effective against ± 1 embedding.

Strategy Q1 (the single black cross at dimensionality $12,753$ in Figure 5.2.3) does not optimize w.r.t. the quantization factor q, and thus it is not surprising that its performance is generally inferior to the performance of the equally-dimensional BEST-q-CLASS strategy with all 39 merged submodels. The loss is however rather small (and there is almost no loss for ± 1 embedding). Additionally, Q1 allows the steganalyst to reduce the feature extraction time roughly to 1/3 as only 39 submodels with $q = 1c$ (out of 106) need to be calculated.

Finally, it is rather interesting that at this payload (0.4 bpp) the EA algorithm is less secure than the simple non-adaptive ± 1 embedding.

5.3 Testing the full framework

The purpose of this section is to test the proposed framework in a way it is customary in research works on steganalysis. In particular, for each split of BOSSbase into 8,074 training images and 1000 testing images and for each payload (0.05, 0.1, 0.2, 0.3, and 0.4 bpp) and stego method, we first use the strategy BEST-q-CLASS to assemble the feature space (from the full rich model) as well as the final steganalyzer. We do so by using only the training set. Note that it is entirely possible that the submodel ranking is slightly different on each split. Once the steganalyzer is trained, we evaluate its detection performance in terms of the testing error P_{E} (2.6.3) achieved on the testing set as a function of the payload expressed in bpp.

In Figure 5.3.1, we plot median values of P_{E} taken over ten independent splits. Median absolute deviation (MAD) values are also included in the figure, together with detection errors for the CDF set [82] implemented with a Gaussian SVM – the state-of-the-art approach before introducing rich models and ensemble classifiers.[4] The figure contains the results for several feature spaces depending on how many submodels in the BEST-q-CLASS strategy are used; TOPn means that the first n submodels were used.

The results confirm that HUGO is by far the best algorithm of all three stegoschemes capable of hiding the payload 0.05 bpp with $P_{\mathrm{E}} \approx 0.42$. Surprisingly, the security of the EA algorithm is comparable with that of ± 1 embedding for payloads larger than 0.3 bpp. We observed that at higher payloads the EA algorithm loses much of its adaptivity and embeds with higher change rate than ± 1 embedding due to its less sophisticated syndrome coding. For smaller payloads, the EA algorithm is only slightly more secure than ± 1 embedding. Overall, the detection of both adaptive stego methods benefits more from the rich model than ± 1 embedding, which is to be expected and was commented upon already in Section 5.2.1.

[4] To save on processing time, we report the results for the CDF set with a G-SVM only for a single split as the variations over different splits are rather small and similar to those of the ensemble.

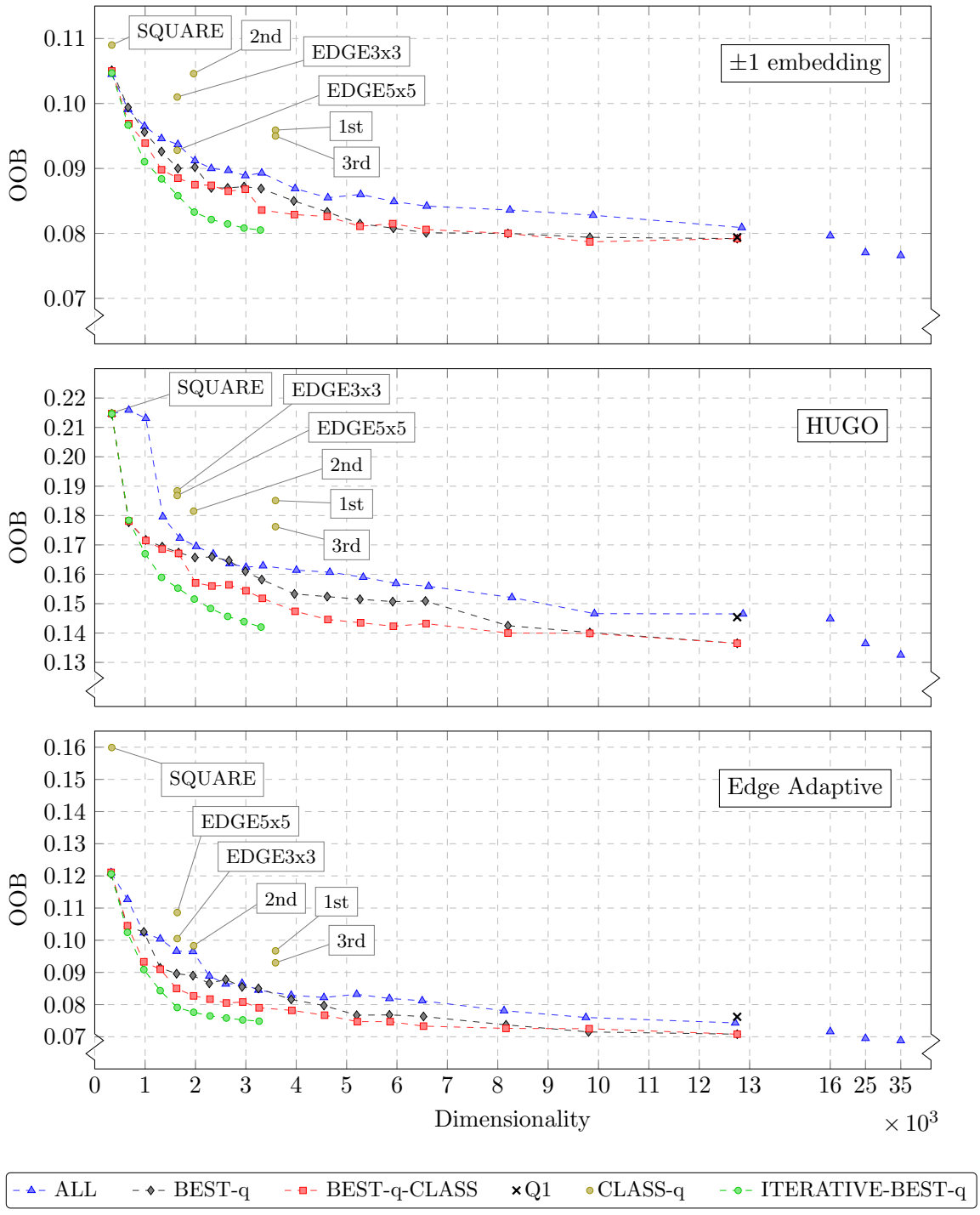
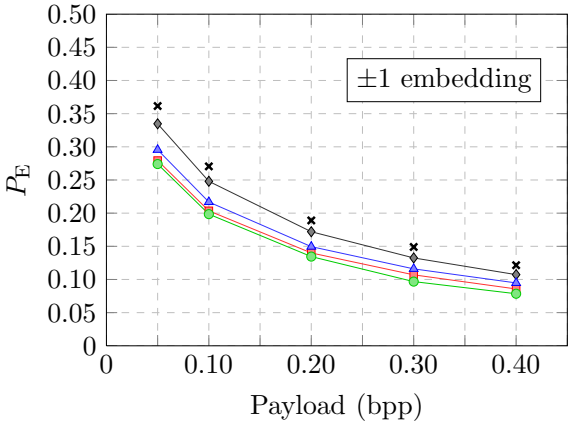


Figure 5.2.3: Performance-to-model dimensionality trade-off for five different submodel selection strategies for three algorithms and a fixed relative payload of 0.4 bpp. The performance is reported in terms of OOB error estimates. The last three tics on the x axis for strategy ALL are not drawn to scale. The last point corresponds to a model in which all quantized versions of all 106 submodels are merged.

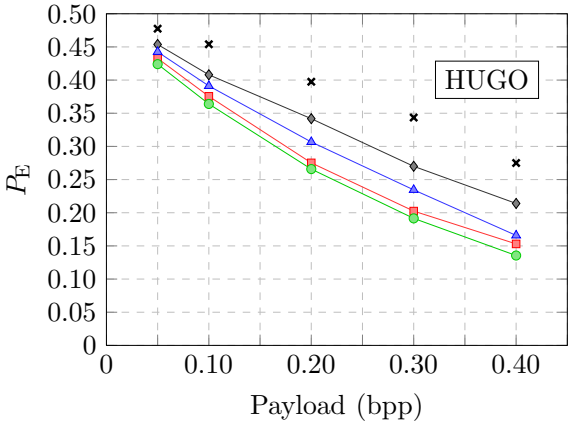
±1 embedding

payload (bpp)	CDF (G-SVM) (single run)	TOP 39 MED	TOP 39 MAD
0.05	0.3615	0.2740	0.0065
0.10	0.2705	0.1985	0.0057
0.20	0.1890	0.1345	0.0035
0.30	0.1490	0.0968	0.0038
0.40	0.1215	0.0785	0.0035



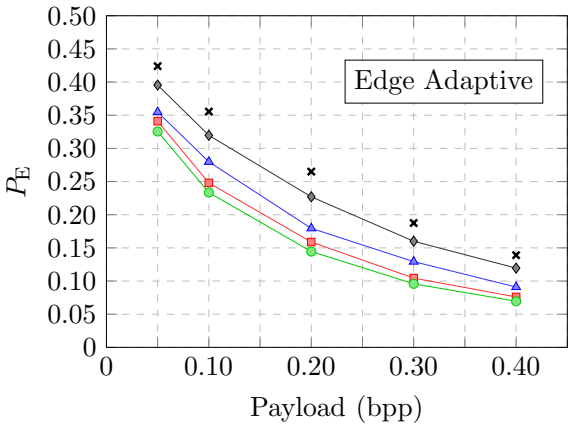
HUGO

payload (bpp)	CDF (G-SVM) (single run)	TOP 39 MED	TOP 39 MAD
0.05	0.4775	0.4240	0.0045
0.10	0.4540	0.3640	0.0023
0.20	0.3975	0.2658	0.0053
0.30	0.3435	0.1915	0.0033
0.40	0.2750	0.1355	0.0035



Edge Adaptive

payload (bpp)	CDF (G-SVM) (single run)	TOP 39 MED	TOP 39 MAD
0.05	0.4240	0.3255	0.0028
0.10	0.3555	0.2335	0.0067
0.20	0.2650	0.1445	0.0037
0.30	0.1875	0.0958	0.0038
0.40	0.1390	0.0695	0.0020



CDF	...	dimension $d = 1234$
TOP 39	...	dimension $d = 12753$

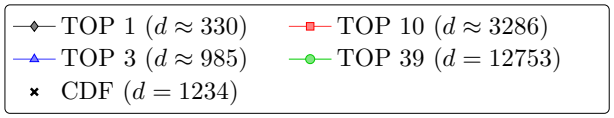


Figure 5.3.1: Detectability of three stego algorithms as a function of payload for several rich models. We plot median values over ten database splits into 8074/1000 training/testing images. The models as well as the classifiers were constructed for each split. The model assembly strategy was BEST-q-CLASS. The tables on the left contain the numerical values and a comparison with a classifier implemented using Gaussian SVM with the CDF set.

The proposed detectors provide a substantial improvement in detection accuracy over the 1234-dimensional CDF set with a Gaussian SVM even when the smallest model (TOP1 with dimensionality slightly above 300) is used. This improvement is again much higher for the two adaptive stego algorithms. With regards to the more recent publications on detection of HUGO, we note that the results of [54] were reported for HUGO implemented with $T = 90$, which introduces artifacts that make the steganalysis significantly more accurate and thus incomparable with the HUGO algorithm run with $T = 255$ tested here (see Chapter 4.3.3 and [80] for more details). Even though the attacks on HUGO reported in [50, 49, 78, 54] did not explicitly utilize the above-mentioned weakness, they, too are likely affected by the weakness and are thus not directly comparable. Having said this, the best results of [50] on HUGO (with $T = 90$) achieved with model dimensionality of 33,930 can now be matched with our rich model with dimensionality 30–100 times smaller. The decrease in the detection error P_{E} ranges from roughly 6% (for payload 0.1 bpp) to about 3% for payload 0.4 bpp. For ± 1 embedding, the improvement is smaller and ranges from 1–2%.

5.4 Using Gaussian SVM with selected submodels

The assembling strategies studied in Experiment 2 in Section 5.2.2 could be viewed independently of the final classifier design. In other words, it is certainly possible to use the speed and convenience of the ensemble to assemble the low-dimensional model, for example through the ITERATIVE-BEST-q strategy, and then use the resulting feature space for training of a different machine-learning tool that may provide a better separation between classes when a highly non-linear boundary exists that may not be well captured by the ensemble equipped with linear base learners. In fact, we observed that features built as co-occurrences of neighboring noise residuals often lead to non-linear boundaries that are better captured by Gaussian SVMs (G-SVMs) than the ensemble. This has already been observed in our previous work [50] and is confirmed for our rich model as well.[5]

To demonstrate the potential of this idea, we decided to include one more experiment. We took the feature spaces assembled using the strategy ITERATIVE-BEST-q with ten submodels (dimension approximately 3,300) and trained a G-SVM for all three algorithms using the same experimental setup as in the experiments of Figure 5.3.1. This was the largest model we could afford to use with a G-SVM given our computing resources. Calculating the median detection error over ten splits, in Table 5.3 we compare the results with the detection error of classifiers implemented as ensembles using the 12,753-dimensional TOP39 model. We only show the results for the 0.4 bpp payload as carrying out these types of experiments under our experimental setting (feature dimensionality and training set size) is rather expensive with a G-SVM. Interestingly, the smaller model with a G-SVM as the final classifier provided better detection results. This could be attributed to the better ability of a G-SVM to learn very complex (non-linear) decision boundaries caused by inhomogeneity of the BOSSbase image database consisting of 7 different cover sources.

However, the improvement of G-SVM is only by 0.5–1% over all three steganographic methods, in terms of the median testing error, with a similar level of statistical variability over the splits. More importantly, the running time of a G-SVM classifier with 3,300-dimensional features was on average 30–90 times higher than the running time of the ensemble classifier with 12,753-dimensional features, as reported in Table 5.4. The measured running times correspond to the full training and testing, including the parameter-search procedures of both types of classifiers. In case of the ensemble, this is the search for the optimal value of d_{sub} through OOB estimation (see Chapter 3.2), and in case of a G-SVM it is a five-fold cross-validation search for the optimal hyper-parameters – the cost parameter C and the kernel width γ. It was carried out on the multiplicative grid

$$\mathcal{G}_C \times \mathcal{G}_\gamma,\ \mathcal{G}_C = \{10^a\},\ a \in \{0, \dots, 4\},\ \mathcal{G}_\gamma = \left\{ \frac{1}{d} \cdot 2^b \right\},\ b \in \{-4, \dots, 3\}, \tag{5.4.1}$$

[5] In contrast, in the JPEG domain co-occurrences between quantized DCT coefficients appear to react in a more linear fashion to embedding, causing the ensemble to perform equally well as G-SVMs [78, 81].

Algorithm	Ensemble		G-SVM	
	MED	MAD	MED	MAD
±1 Embedding	0.0785	0.0035	0.0683	0.0042
HUGO	0.1355	0.0035	0.1310	0.0065
EA	0.0695	0.0020	0.0643	0.0030

Table 5.3: Detection error P_{E} for three algorithms for payload 0.4 bpp when the ensemble is used with the rich 12,753-dimensional TOP39 model and when a G-SVM is combined with the $\sim 3,300$-dimensional best ITERATIVE-BEST-q model. The reported numbers are achieved over ten splits of BOSSbase.

Algorithm	Ensemble	G-SVM
±1 Embedding	1 hr 20 min	4 days 22 hr 37 min
HUGO	4 hr 35 min	8 days 15 hr 31 min
EA	3 hr 09 min	3 days 23 hr 50 min

Table 5.4: The average running time (for the training and testing together) of the experiments in Table 5.3 if executed on a single computer with the AMD Opteron 275 processor running at 2.2 GHz.

where d is the feature space dimensionality. We used our Matlab implementation of the ensemble classifier[6] and the publicly available package LIBSVM [21] (with manually implemented cross-validation, see Appendix C.3.1 of this dissertation) to conduct the G-SVM experiments.

5.5 Summary

In this chapter, we constructed a rich model for steganalysis of digital images stored in the pixel format – rich in the sense that it considers numerous qualitatively different relationships among pixels. The proposed Spatial domain Rich Model (SRM), thoroughly described in Section 5.1, is based on 39 types of linear filters that are shown in Figure 5.1.2. Considering also their differently quantized versions, the total number of submodels of SRM is equal to 106 and its dimension is $34,671$. The proposed SRM heavily utilizes symmetries of natural images to compactify its submodels – the philosophy behind building the rich model was to maximize the diversity and statistical significance of its components. Note that this is quite different from the model used in HUGO, where the authors simply increased a truncation threshold to obtain a high-dimensional model.

In Section 5.2, we ask whether it is possible to find a small subset of SRM that is responsible for majority of its detection accuracy. It turns out that there is no subset that would be universally good across different stegoschemes. However, this task becomes feasible for a fixed steganographic channel. This was demonstrated by implementing several feature selection strategies, applied to three different stego algorithms operating in the spatial domain: ±1 embedding, HUGO, and an edge-adaptive (EA) method by Luo *et al.* [94]. In particular, using the ITERATIVE-BEST-q selection strategy that sequentially selects submodels in a greedy manner to minimize the error estimate, we were able to achieve accuracy only slightly inferior to the one of the full SRM with as few as $\sim 3,300$ features (10 selected submodels).

Needless to say, the steganalysis using the high-dimensional SRM, as well as the experimental comparison of different feature-selection techniques, would not be possible without the use of the ensemble classifier introduced in Chapter 3 because of its scalability and low complexity. In fact, the bottleneck now becomes feature extraction rather than the classification itself.

[6] Available at `http://dde.binghamton.edu/download/ensemble`.

Rich-model-based steganalysis with the ensemble classifier significantly outperforms previously-proposed detectors. This was demonstrated in Figure 5.3.1 where we steganalyzed all three spatial domain steganographic techniques at a wide range of relative payloads. The rich-model approach was especially effective on the two adaptive methods (HUGO and EA), where the improvement over prior art (CDF features with Gaussian SVM) was often 10 – 15% of the detection accuracy. This is because adaptive methods place embedding changes in hard-to-model regions of images where the rich model better discerns the embedding changes.

Besides steganalysis, the rich model could be used for steganography as well – by endowing the model with an appropriate distortion function using, e.g., the method described in [35]. We hypothesize, however, that steganographic methods based on minimizing distortion in a rich model space may no longer be able to embed large payloads undetectably as it will become increasingly harder to preserve a large number of statistically significant quantities. This statement stems from an observation made in Experiment 2 in Section 5.2.2, namely that submodels built from first-order differences among pixels are able to detect HUGO relatively reliably despite the fact that its distortion function minimizes perturbations to joint statistics built from such differences.

We expect that the proposed rich model might find applications beyond steganography and steganalysis in related fields, such as digital forensics, for problems dealing with imaging hardware identification, media integrity, processing history recovery, forgery detection, and authentication. A similar framework based on rich models can likely be adopted for other media types, including audio and video signals.

Chapter 6

JPEG domain rich model (JRM)

In the previous chapter, we built a rich model of digital images stored in a raster format (BMP, PNG, PGM, etc.). That is a very intuitive and easy-to-interpret representation in which images are simply matrices of integers representing individual pixel values. As such, it was a good starting point for explaining the concepts of a rich model, because its individual components (submodels) were interpretable. For example, submodels constructed from the second-order residuals assumed a local linearity of the image, while the third-order residuals considered a locally-quadratic model. A larger quantization factor q allowed us to better capture image statistics around edges and in textured areas.

However, the most common image format for storing and transmitting photographs over the Internet is JPEG, the default output of the vast majority of today's digital cameras. JPEG offers an adjustable degree of lossy compression, allowing the user to select a desired trade-off between the image quality and its storage size. The process of JPEG compression could be summarized in the following steps. First, the pixel values are transformed into the frequency domain through the Discrete Cosine Transform (DCT). The resulting coefficients are then quantized, rounded to integers, losslessly Huffman-coded, and stored as a JPEG file. Since DCT coefficients could be easily obtained from the JPEG file through entropy decoding, we will use the term JPEG image and DCT plane interchangeably.

The popularity of JPEG format makes it an appealing cover source for steganographic communication. Numerous steganographic algorithms hiding information into DCT coefficients have been proposed in the past, examples of which are [132, 116, 72, 118, 115], and steganalysis of JPEG images is an area of active research. In the light of the encouraging results of the previous chapter, it is only natural to ask whether a similar methodology can be applied here as well, i.e. whether we could improve upon existing detectors by constructing a rich model in JPEG domain.

In this chapter, we will show that it is indeed possible – we construct a rich model of DCT coefficients for steganalysis of JPEG images and experimentally demonstrate its superior performance over previous art. We will abbreviate the proposed model JRM, mnemonic for JPEG Rich Model. We note that even though conceptually similar to the construction of the spatial domain rich model, the development of JRM needs to be carried out differently, because DCT coefficients have fundamentally different statistical properties than pixel values in the spatial domain. In particular, coefficients in different DCT modes (positions within 8×8 blocks) may be viewed as different noise residuals, and thus it makes sense to form sample statistics *directly* from their values. Another difference is the concept of Cartesian calibration discussed in Chapter 4.2 of this dissertation – a technique that was absent in Chapter 5 but will be utilized here. We will denote the Cartesian-calibrated JRM as CC-JRM.

The chapter is organized as follows. The formal description of the proposed CC-JRM is provided in Section 6.1. In Section 6.2, its ability to detect a wide range of qualitatively different embedding

schemes is demonstrated on six modern steganographic schemes hiding data into JPEG images: nsF5 [51], MBS [116], YASS [118], MME [72], BCH, and BCHopt [115] (see Appendix A for details on these algorithms). The proposed CC-JRM clearly outperforms prior art, including both low- and high-dimensional feature sets.

Section 6.3 is reserved for a series of investigative experiments revealing interesting insight and interpretations about the individual components of the CC-JRM and their contributions to the detection of individual stego schemes. We also apply the ITERATIVE-BEST feature selection strategy introduced in Chapter 5.2.2 and reveal an interesting property of Cartesian calibration in high-dimensional feature spaces.

The chapter is summarized in Section 6.4.

6.1 Building the rich model

A JPEG image consists of 64 parallel channels formed by DCT modes which exhibit complex but short-distance dependencies of two types – frequency (intra-block) and spatial (inter-block). The former relates to the relationship among coefficients with similar frequency within the same 8×8 block while the latter refers to the relationship across different blocks. Although the statistics of neighboring DCT coefficients have been used as models in the past many times [39, 123, 107, 22, 93, 92], the need to keep the model dimensionality low for the subsequent classifier training usually limited the model scope to co-occurrence matrices (or transition probability matrices) constructed from *all* coefficients in the DCT plane. Thus, despite their very different statistical nature, all DCT modes were treated equally.

Our proposed rich model consists of several qualitatively different parts. First, in the lines of our previously proposed $\mathcal{CF}^*$ features [81], we model individual DCT modes *separately,* collect many of these submodels and put them together. They will be naturally diverse since they capture dependencies among different DCT coefficients. The second part of the proposed JRM is formed as *integral* statistics from the whole DCT plane. The increased statistical power enables us to extend the range of co-occurrence features and therefore cover a different spectrum of dependencies than the mode-specific features from the first part. The features of both parts are further diversified by modeling not only DCT coefficients themselves, but also their *differences* calculated in different directions.

6.1.1 Notation and definitions

Quantized DCT coefficients of a JPEG image of dimensions $n_1 \times n_2$ will be represented by a matrix $\mathbf{D} \in \mathbb{Z}^{n_1 \times n_2}$. Let $D_{xy}^{(i,j)}$ denote the (x,y)th DCT coefficient in the (i,j)th 8×8 block, $(x,y) \in \{0,\dots,7\}^2$, $i = 1,\dots,\lceil n_1/8 \rceil$, $j = 1,\dots,\lceil n_2/8 \rceil$. Alternatively, we may access individual elements of $\mathbf{D}$ as D_{ij}, $i = 1,\dots,n_1$, $j = 1,\dots,n_2$. We define the following matrices:

$$\mathbf{A}^{\times} \ \dots \ A_{ij}^{\times} = |D_{ij}|, \ i = 1,\dots,n_1, \ j = 1,\dots,n_2, \tag{6.1.1}$$

$$\mathbf{A}^{\rightarrow} \ \dots \ A_{ij}^{\rightarrow} = |D_{ij}| - |D_{i,j+1}|, \ i = 1,\dots,n_1, \ j = 1,\dots,n_2 - 1, \tag{6.1.2}$$

$$\mathbf{A}^{\downarrow} \ \dots \ A_{ij}^{\downarrow} = |D_{ij}| - |D_{i+1,j}|, \ i = 1,\dots,n_1 - 1, \ j = 1,\dots,n_2, \tag{6.1.3}$$

$$\mathbf{A}^{\searrow} \ \dots \ A_{ij}^{\searrow} = |D_{ij}| - |D_{i+1,j+1}|, \ i = 1,\dots,n_1 - 1, \ j = 1,\dots,n_2 - 1, \tag{6.1.4}$$

$$\mathbf{A}^{\rightrightarrows} \ \dots \ A_{ij}^{\rightrightarrows} = |D_{ij}| - |D_{i,j+8}|, \ i = 1,\dots,n_1, \ j = 1,\dots,n_2 - 8, \tag{6.1.5}$$

$$\mathbf{A}^{\downdownarrows} \ \dots \ A_{ij}^{\downdownarrows} = |D_{ij}| - |D_{i+8,j}|, \ i = 1,\dots,n_1 - 8, \ j = 1,\dots,n_2. \tag{6.1.6}$$

Matrix $\mathbf{A}^{\times}$ consists of the absolute values of DCT coefficients, matrices $\mathbf{A}^{\rightarrow}, \mathbf{A}^{\downarrow}, \mathbf{A}^{\searrow}$ are obtained as intra-block differences, and $\mathbf{A}^{\rightrightarrows}, \mathbf{A}^{\downdownarrows}$ represent inter-block differences. Individual submodels of

the proposed JRM will be formed as 2D co-occurrence matrices calculated from the coefficients of matrices $\mathbf{A}^\star, \star \in \{\times, \rightarrow, \downarrow, \searrow, \rightrightarrows, \downdownarrows\}$, positioned in DCT modes (x, y) and $(x+\Delta x, y+\Delta y)$. Formally, $\mathbf{C}^\star_T(x, y, \Delta x, \Delta y)$, $\star \in \{\times, \rightarrow, \downarrow, \searrow, \rightrightarrows, \downdownarrows\}$, are $(2T+1)^2$-dimensional matrices with elements

$$C^\star_{kl}(x, y, \Delta x, \Delta y) = \frac{1}{Z} \sum_{i,j} \left| \left\{ \tilde{A}^{(i,j)}_{xy} \middle| \tilde{\mathbf{A}} = \mathrm{trunc}_T(\mathbf{A}^\star);\ \tilde{A}^{(i,j)}_{xy} = k;\ \tilde{A}^{(i,j)}_{x+\Delta x, y+\Delta y} = l \right\} \right|, \tag{6.1.7}$$

where the normalization constant Z ensures that $\sum_{k,l} C^\star_{kl} = 1$ and $\mathrm{trunc}_T(\cdot)$ is an element-wise truncation operator defined in Chapter 1.5 by (1.5.1). In definition (6.1.7), we do not constrain Δx and Δy and allow $(x + \Delta x, y + \Delta y)$ to be out of the range $\{0, \ldots, 7\}^2$ to more easily describe co-occurrences for inter-block coefficient pairs, e.g., $\tilde{A}^{(i,j)}_{x+8,y} \equiv \tilde{A}^{(i+1,j)}_{xy}$.

Assuming the statistics of natural images do not change after mirroring about the main diagonal, the symmetry of DCT basis functions w.r.t. the 8×8 block diagonal allows us to replace matrices $\mathbf{C}^\star_T$ with the more robust

$$\bar{\mathbf{C}}^\times_T(x, y, \Delta x, \Delta y) \triangleq \frac{1}{2} \left(\mathbf{C}^\times_T(x, y, \Delta x, \Delta y) + \mathbf{C}^\times_T(y, x, \Delta y, \Delta x) \right), \tag{6.1.8}$$

$$\bar{\mathbf{C}}^\rightarrow_T(x, y, \Delta x, \Delta y) \triangleq \frac{1}{2} \left(\mathbf{C}^\rightarrow_T(x, y, \Delta x, \Delta y) + \mathbf{C}^\downarrow_T(y, x, \Delta y, \Delta x) \right), \tag{6.1.9}$$

$$\bar{\mathbf{C}}^\rightrightarrows_T(x, y, \Delta x, \Delta y) \triangleq \frac{1}{2} \left(\mathbf{C}^\rightrightarrows_T(x, y, \Delta x, \Delta y) + \mathbf{C}^\downdownarrows_T(y, x, \Delta y, \Delta x) \right), \tag{6.1.10}$$

$$\bar{\mathbf{C}}^\searrow_T(x, y, \Delta x, \Delta y) \triangleq \frac{1}{2} \left(\mathbf{C}^\searrow_T(x, y, \Delta x, \Delta y) + \mathbf{C}^\searrow_T(y, x, \Delta y, \Delta x) \right). \tag{6.1.11}$$

Because the coefficients in $\mathbf{A}^\times$ are non-negative, most of the bins of $\bar{\mathbf{C}}^\times_T$ are zeros and its true dimensionality is only $(T+1)^2$. The difference-based co-occurrences $\bar{\mathbf{C}}^\star_T$, $\star \in \{\rightarrow, \searrow, \rightrightarrows\}$, are generally nonzero, however, we can additionally utilize their *sign symmetry* ($C^\star_{kl} \approx C^\star_{-k,-l}$) and define $\hat{\mathbf{C}}^\star_T$ with elements

$$\hat{C}^\star_{kl} = \frac{1}{2} \left(\bar{C}^\star_{kl} + \bar{C}^\star_{-k,-l} \right). \tag{6.1.12}$$

The redundant portions of $\hat{\mathbf{C}}^\star_T$ can be removed to obtain the final form of the difference-based co-occurrences of dimensionality $\frac{1}{2}(2T+1)^2 + \frac{1}{2}$, which we denote again $\hat{\mathbf{C}}^\star_T(x, y, \Delta x, \Delta y)$, $\star \in \{\rightarrow, \searrow, \rightrightarrows\}$. The rich model will be constructed only using the most compact forms: $\bar{\mathbf{C}}^\times_T$, $\hat{\mathbf{C}}^\rightarrow_T$, $\hat{\mathbf{C}}^\searrow_T$, and $\hat{\mathbf{C}}^\rightrightarrows_T$.

We note that the co-occurrences $\bar{\mathbf{C}}^\times_T$ evolved from the $\mathcal{F}^*$ feature set proposed in [81]. The difference is that $\mathcal{F}^*$ does not take absolute values before forming co-occurrences. Taking absolute values reduces dimensionality and makes the features more robust; it could be seen as another type of symmetrization. Later in Section 6.2, we compare the performance of the proposed rich model with $\mathcal{CF}^*$, the Cartesian-calibrated $\mathcal{F}^*$set [81].

6.1.2 DCT-mode specific components of JRM

Depending on the mutual position of the DCT modes (x, y) and $(x + \Delta x, y + \Delta y)$, the extracted co-occurrence matrices $\mathbf{C} \in \{\bar{\mathbf{C}}^\times_T, \hat{\mathbf{C}}^\rightarrow_T, \hat{\mathbf{C}}^\searrow_T, \hat{\mathbf{C}}^\rightrightarrows_T\}$ will be grouped into ten qualitatively different submodels:

$$
\begin{aligned}
1.\quad & \mathcal{G}_{\mathrm{h}}(\mathbf{C}) &&= \{\mathbf{C}(x,y,0,1)|0\le x;\ 0\le y;\ x+y\le 5\},\\
2.\quad & \mathcal{G}_{\mathrm{d}}(\mathbf{C}) &&= \{\mathbf{C}(x,y,1,1)|0\le x\le y;\ x+y\le 5\}\cup\{\mathbf{C}(x,y,1,-1)|0\le x<y;\ x+y\le 5\},\\
3.\quad & \mathcal{G}_{\mathrm{oh}}(\mathbf{C}) &&= \{\mathbf{C}(x,y,0,2)|0\le x;\ 0\le y;\ x+y\le 4\},\\
4.\quad & \mathcal{G}_{\mathrm{x}}(\mathbf{C}) &&= \{\mathbf{C}(x,y,y-x,x-y)|0\le x<y;\ x+y\le 5\},\\
5.\quad & \mathcal{G}_{\mathrm{od}}(\mathbf{C}) &&= \{\mathbf{C}(x,y,2,2)|0\le x\le y;\ x+y\le 4\}\cup\{\mathbf{C}(x,y,2,-2)|0\le x<y;\ x+y\le 5\},\\
6.\quad & \mathcal{G}_{\mathrm{km}}(\mathbf{C}) &&= \{\mathbf{C}(x,y,-1,2)|1\le x;\ 0\le y;\ x+y\le 5\},\\
7.\quad & \mathcal{G}_{\mathrm{ih}}(\mathbf{C}) &&= \{\mathbf{C}(x,y,0,8)|0\le x;\ 0\le y;\ x+y\le 5\},\\
8.\quad & \mathcal{G}_{\mathrm{id}}(\mathbf{C}) &&= \{\mathbf{C}(x,y,8,8)|0\le x\le y;\ x+y\le 5\},\\
9.\quad & \mathcal{G}_{\mathrm{im}}(\mathbf{C}) &&= \{\mathbf{C}(x,y,-8,8)|0\le x\le y;\ x+y\le 5\},\\
10.\quad & \mathcal{G}_{\mathrm{ix}}(\mathbf{C}) &&= \{\mathbf{C}(x,y,y-x,x-y+8)|0\le x;\ 0\le y;\ x+y\le 5\}.
\end{aligned}
$$

The first six submodels capture intra-block relationships: $\mathcal{G}_{\mathrm{h}}$ – horizontally (and vertically, after symmetrization) neighboring pairs; $\mathcal{G}_{\mathrm{d}}$ – diagonally and minor-diagonally neighboring pairs; $\mathcal{G}_{\mathrm{oh}}$ – "skip one" horizontally neighboring pairs; $\mathcal{G}_{\mathrm{x}}$ – pairs symmetrically positioned w.r.t. the 8×8 block diagonal; $\mathcal{G}_{\mathrm{od}}$ – "skip one" diagonal and minor-diagonal pairs; $\mathcal{G}_{\mathrm{km}}$ – "knight-move" positioned pairs. The last four submodels capture inter-block relationships between coefficients from neighboring blocks: $\mathcal{G}_{\mathrm{ih}}$ – horizontal neighbors in the same DCT mode; $\mathcal{G}_{\mathrm{id}}$ – diagonal neighbors in the same mode; $\mathcal{G}_{\mathrm{im}}$ – minor-diagonal neighbors in the same mode, $\mathcal{G}_{\mathrm{ix}}$ – horizontal neighbors in modes symmetrically positioned w.r.t. 8×8 block diagonal. The two parts forming $\mathcal{G}_{\mathrm{d}}$ and $\mathcal{G}_{\mathrm{od}}$ were grouped together to give all submodels roughly the same dimensionality.

Since all ten groups of submodels are constructed for $\mathbf{C}\in\{\bar{\mathbf{C}}_3^{\times},\hat{\mathbf{C}}_2^{\rightarrow},\hat{\mathbf{C}}_2^{\searrow},\hat{\mathbf{C}}_2^{\rightrightarrows}\}$, 40 DCT-mode specific submodels are obtained in total. For co-occurrences of absolute values of DCT coefficients, we fixed $T=3$ yielding the dimensionality of a single matrix $\bar{\mathbf{C}}_3^{\times}$ equal to 16. For difference-based co-occurrences, we fixed $T=2$ to obtain a similar dimensionality of 13. Larger values of T would result in many underpopulated bins, especially for smaller images. A tabular listing of all introduced submodels, including their total dimensionalities, is shown in Figure 6.1.1.

6.1.3 Integral components of JRM

The mode-specific submodels introduced in Section 6.1.2 give the rich model a finer "granularity" but at the price of utilizing only a small portion of the DCT plane at a time. In order not to lose the integral statistical power of the whole DCT plane and to cover a larger range of DCT coefficients, we now finalize the rich model by supplementing additional co-occurrence matrices that are integrated over all DCT modes. We do so for both the co-occurrences of absolute values of DCT coefficients, $\bar{\mathbf{C}}_T^{\times}$, and their differences, $\hat{\mathbf{C}}_T^{\star}$, $\star\in\{\rightarrow,\downarrow,\searrow,\rightrightarrows,\downdownarrows\}$. As the integral bins are better populated than DCT-mode specific bins, we increase T to 5. The integral submodels are defined as follows:

1. $\mathcal{I}^{\times}=\left\{\sum_{x,y}\bar{\mathbf{C}}_5^{\times}(x,y,\Delta x,\Delta y)\middle|[\Delta x,\Delta y]\in\{(0,1),(1,1),(1,-1),(0,8),(8,8)\}\right\}$,
2. $\mathcal{I}_{\mathrm{f}}^{\star}=\left\{\sum_{x,y}\hat{\mathbf{C}}_5^{\star}(x,y,\Delta x,\Delta y)\middle|[\Delta x,\Delta y]\in\{(0,1),(1,0),(1,1),(1,-1)\}\right\}$, $\star\in\{\rightarrow,\downarrow,\searrow,\rightrightarrows,\downdownarrows\}$,
3. $\mathcal{I}_{\mathrm{s}}^{\star}=\left\{\sum_{x,y}\hat{\mathbf{C}}_5^{\star}(x,y,\Delta x,\Delta y)\middle|[\Delta x,\Delta y]\in\{(0,8),(8,0),(8,8),(8,-8)\}\right\}$, $\star\in\{\rightarrow,\downarrow,\searrow,\rightrightarrows,\downdownarrows\}$,

For intra-block pairs, the summation in the above definitions is always over all DCT modes $(x,y)\in\{0,\ldots,7\}^2$ such that both (x,y) and $(x+\Delta x,y+\Delta y)$ lie within the same 8×8 block. A similar constraint applies to the inter-block matrices whenever the indices would end up outside of the DCT array. DC modes are omitted in all definitions. The submodel $\mathcal{I}^{\times}$ covers both the spatial (inter-block) and frequency (intra-block) dependencies, and can be seen as an extension of feature sets $absNJ_1$ and $absNJ_2$ by Liu [92]. The difference-based submodels bear similarity to the Markov

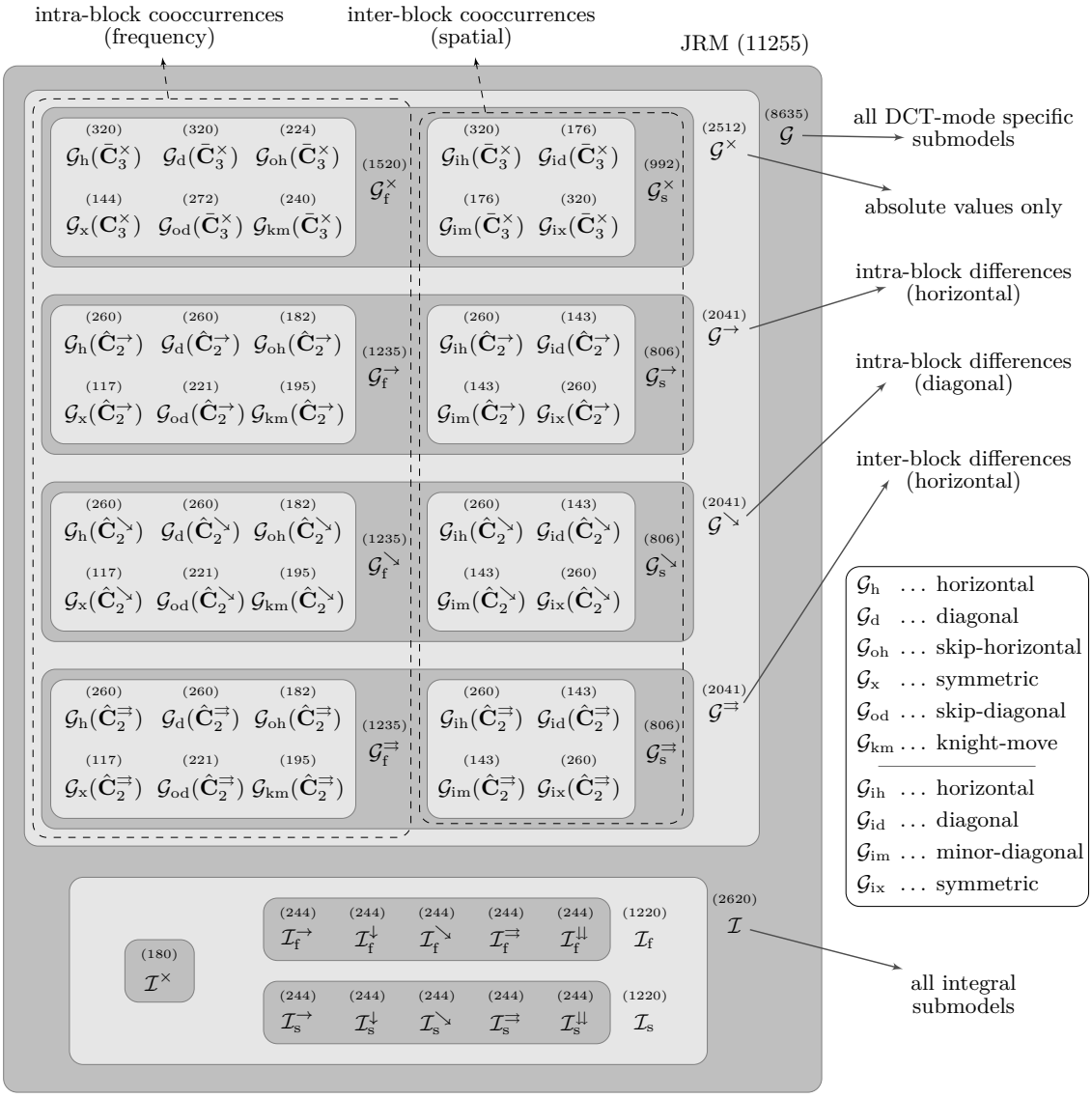


Figure 6.1.1: The proposed JPEG rich model and its decomposition into individual subgroups and submodels. The numbers denote the dimensionalities of the corresponding sets. Cartesian calibration doubles all shown values.

features proposed in [22], where the authors also utilized inter- and intra-block differences between absolute values of DCT coefficients. In order to obtain similar dimensionalities, the co-occurrences calculated from differences were divided into two distinct sets, capturing frequency ($\mathcal{I}_f^\star$) and spatial ($\mathcal{I}_s^\star$) dependencies separately.[1]

The union of DCT-mode specific submodels with the integral submodels form the JPEG domain rich model we propose. Its total dimensionality is 11,255. In order to improve the performance, we apply the Cartesian calibration [77] (Chapter 4.2) which doubles the dimensionality to 22,510. The structure of the entire JRM appears in Figure 6.1.1.

[1]Note that some of the submodels capture quite complex statistical dependencies among DCT coefficients. For example, $\mathcal{D}_f^{\rightrightarrows}$ combines *inter*-block differences with *intra*-block co-occurrences.

6.2 Comparison to prior art

To demonstrate the power of the proposed JPEG rich model, we steganalyze six modern steganographic methods: nsF5, MBS, YASS, MME, BCH, and BCHopt. Their description appears in Appendix A.

6.2.1 Performance evaluation

Similarly to the previous chapter, we use the ensemble classifier [81, 78] for all experiments as it enables fast training in high-dimensional feature spaces and its performance on low-dimensional feature sets is comparable to the much more complex SVMs [81]. The classifier is described in details in Chapter 3 of this dissertation.

All experiments were conducted on the CAMERA image database that is described in Appendix B, and has been already used in Chapters 3 and 4. The cover images for MME were created using a Java JPEG encoder, which is the same compressor that is incorporated in the MME implementation we used.[2] For the rest, we used the Matlab's function `imwrite`. The JPEG quality factor was fixed to 75 in both cases.

For every steganographic method, we created stego images using a range of different payload sizes expressed in terms of bits per nonzero AC DCT coefficient (bpac) and trained a separate classifier to detect each of them. The performance is evaluated using the median value of P_{E} (2.6.3) over ten random 50/50 splits of the database into training and testing set and will be denoted as $\bar{P}_{\text{E}}$.

We compare the steganalysis performance of the following feature spaces (models); the numbers in brackets denote their dimensionality:

- CHEN (486) = Markov features utilizing both intra- and inter-block dependencies [22],
- CC-CHEN (972) = CHEN features improved by Cartesian calibration [77],
- LIU (216) = the union of *diff-absNJ-ratio* and *ref-diff-absNJ* features published in [92],
- CC-PEV (548) = Cartesian-calibrated PEV feature set [107] (see also Chapter 4.2.5),
- CDF (1,234) = CC-PEV features expanded by spatial-domain SPAM features [104],
- CC-C300 (48,600) = the high-dimensional feature space proposed in [78],
- $\mathcal{CF}^*$ (7,850) = compact rich model for DCT domain proposed in [81],
- JRM (11,255) = our rich model proposed in this chapter, without calibration,
- CC-JRM (22,510) = Cartesian-calibrated JRM,
- J+SRM (35,263) = the union of CC-JRM and the Spatial-domain Rich Model (SRM) proposed in Chapter 5. We take the 39 submodels of SRM that were created with a fixed quantization $q = 1c$. This is equivalent to the Q1 feature selection strategy introduced in Chapter 5.2.2.

The resulting errors $\bar{P}_{\text{E}}$ are reported in Table 6.1. The proposed CC-JRM delivers the best performance among all feature sets that are extracted directly from the DCT domain, across all tested steganographic methods and all payloads. Adding the spatial-domain rich model SRM (Q1) further improves the performance and delivers the overall best results.

Algorithm	bpac	CHEN (486)	CC-CHEN (972)	LIU (216)	CC-PEV (548)	CDF (1,234)	CC-C300 (48,600)	$\mathcal{CF}^*$ (7,850)	JRM (11,255)	CC-JRM (22,510)	J+SRM (35,263)
nsF5	.050	.4153	.3816	.3377	.3690	.3594	.3722	.3377	.3407	.3298	.3146
	.100	.3097	.2470	.1732	.2239	.2020	.2207	.1737	.1782	.1616	.1375
	.150	.2094	.1393	.0706	.1171	.0906	.1127	.0720	.0793	.0663	.0468
	.200	.1345	.0708	.0273	.0549	.0360	.0486	.0273	.0338	.0255	.0150
MBS	.010	.4070	.3962	.3826	.3876	.3786	.4038	.3710	.3478	.3414	.3260
	.020	.3178	.2962	.2780	.2827	.2684	.3120	.2560	.2156	.2122	.1832
	.030	.2395	.2100	.1925	.1965	.1795	.2241	.1684	.1266	.1195	.0983
	.040	.1770	.1437	.1288	.1298	.1135	.1594	.1087	.0751	.0670	.0494
	.050	.1243	.0946	.0812	.0833	.0704	.1176	.0684	.0427	.0373	.0282
YASS (12)	.077	.2009	.1825	.2324	.2279	.1268	.0930	.0532	.0324	.0303	.0173
YASS (11)	.114	.1989	.1585	.2118	.1573	.0718	.0701	.0437	.0349	.0227	.0111
YASS (8)	.138	.2520	.1911	.1886	.1827	.0742	.0500	.0271	.0287	.0178	.0104
YASS (10)	.159	.2334	.1476	.1793	.1341	.0507	.0370	.0164	.0210	.0103	.0054
YASS (3)	.187	.1277	.0876	.1301	.0723	.0224	.0350	.0146	.0165	.0081	.0045
MME	.050	.4678	.4546	.4479	.4492	.4340	.4427	.4443	.4424	.4307	.4194
	.100	.3001	.2611	.2574	.2613	.2501	.3026	.2466	.2286	.2091	.1891
	.150	.2165	.1735	.1677	.1721	.1586	.2299	.1608	.1404	.1221	.1027
	.200	.0217	.0104	.0127	.0127	.0124	.0726	.0153	.0112	.0080	.0059
BCH	.100	.4599	.4496	.4448	.4426	.4390	.4497	.4290	.4305	.4229	.4060
	.200	.3594	.3124	.3087	.2974	.2752	.2958	.2629	.2707	.2369	.1946
	.300	.1383	.0889	.0862	.0779	.0697	.0912	.0663	.0715	.0536	.0390
BCHopt	.100	.4726	.4683	.4558	.4618	.4595	.4684	.4550	.4515	.4480	.4306
	.200	.4032	.3712	.3583	.3548	.3368	.3517	.3265	.3253	.3030	.2582
	.300	.2400	.1711	.1719	.1605	.1356	.1681	.1289	.1389	.1102	.0830

Table 6.1: Median testing error $\bar{P}_{\mathrm{E}}$ for six JPEG steganographic methods using different models. For easier navigation, the gray-level of the background in each row corresponds to the performance of individual feature sets: darker ⇒ better performance (lower error rate).

6.2.2 Discussion

Table 6.1 provides an important insight into the feature-building process and points to the following general guidelines for design of feature spaces for steganalysis, some of them already discussed in Chapter 4:

- *High dimension is not sufficient for good performance.* This is clearly demonstrated by the rather poor performance of the 48,600-dimensional CC-C300 feature set, often outperformed by the significantly lower-dimensional sets LIU, CC-PEV, and CC-CHEN. The failure of CC-C300 could be attributed to its lack of diversity (all co-occurrences are of the same type) and missing symmetrization, which makes the model less robust and unnecessarily high-dimensional.
- *Calibration helps.* The positive effect of calibration has been demonstrated many times in the past, and here we confirm it by comparing the columns CHEN → CC-CHEN and JRM → CC-JRM. Notice that even for the high-dimensional JRM, the improvement may be substantial: 0.2707 → 0.2369 for BCH at 0.2 bpac or 0.1404 → 0.1221 for MME at 0.15 bpac. Moreover, researching alternative ways of calibration may bring additional improvements to feature-based steganalysis. This is indicated by a relatively good performance of the LIU feature set (compared to other low-dimensional sets), which utilizes two novel calibration principles: strengthening the reference statistics by averaging over 63 different image croppings and calibrating by the *ratio* between original and reference features [92].

[2]See Appendix C.1 for a deeper discussion on the importance of this issue.

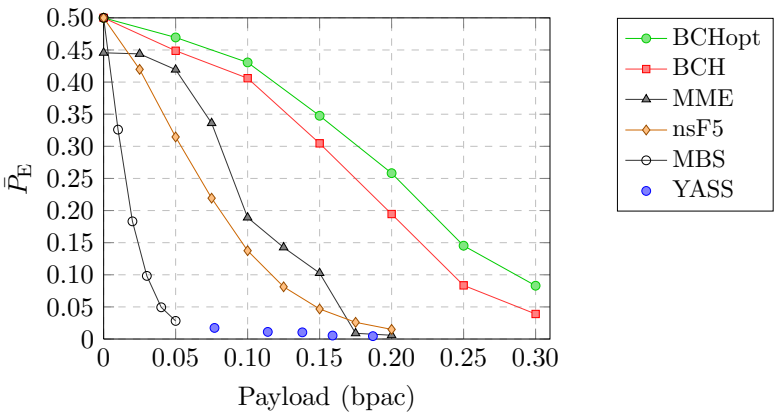


Figure 6.2.1: Testing error $\bar{P}_E$ using the J+SRM feature space (dimension 35,263).

- *Steganalysis benefits from cross-domain models.* By combining CC-PEV with the spatial-domain SPAM features (CDF), sizeable improvement over CC-PEV is apparent across all steganographic methods. The benefit of the multiple-domain approach is also clear from the last column – the union of the CC-JRM and the spatial domain rich model proposed in the previous chapter of this dissertation further markedly improves the performance of CC-JRM and yields the lowest achieved error rates in all cases.
- *Future steganalysis will likely be driven by diverse and compact rich models.* The systematically constructed JPEG rich models $\mathcal{CF}^*$ and JRM/CC-JRM consistently outperform all low-dimensional sets. The superior performance of CC-JRM over $\mathcal{CF}^*$ is due to additional symmetrization, further diversification by co-occurrences of *differences*, and by its new integral components.

6.2.3 Comparison of steganographic methods

In Figure 6.2.1, we compare the performance of all tested stego schemes using the J+SRM feature set. BCHopt is clearly the most secure tested steganographic method, followed by BCH which is its earlier version, without heuristic optimization, see [115] for more details.

MBS and YASS are by far the least secure algorithms. The failure of YASS, already reported for example in [82, 92], suggests that embedding robustly in a different domain may not be the best approach for passive warden steganography as the robustness unavoidably yields significant and thus easily detectable distortion.

The nsF5 algorithm, which does not utilize any side information, is clearly outperformed by all schemes that utilize the knowledge of the uncompressed cover: MME, BCH, and BCHopt. The effect of this type of side information at the sender on steganographic security is, however, not well understood today. In particular, it is not clear how to utilize it in the best possible manner.

Let us conclude this section by commenting on two security artifacts of MME. First, we can see significant jumps in $\bar{P}_E$ around payloads 0.09 and 0.16 bpac, which are due to suboptimal Hamming codes as already reported in [82]. This could be easily remedied by using more sophisticated coding schemes [36]. Second, note that the error at zero payload is $\bar{P}_E \approx 0.45$ rather than random guessing. This is caused by the embedded message header whose size is independent of the message length and which is present in every stego image. We found that in case of MME, this message header is always embedded in the top left corner of the image, in many cases the area of sky, and thus creates statistically detectable traces. Even though this implementation flaw could be easily fixed, it illustrates that even the smallest implementation detail needs to be handled with caution when designing a practical steganographic scheme.

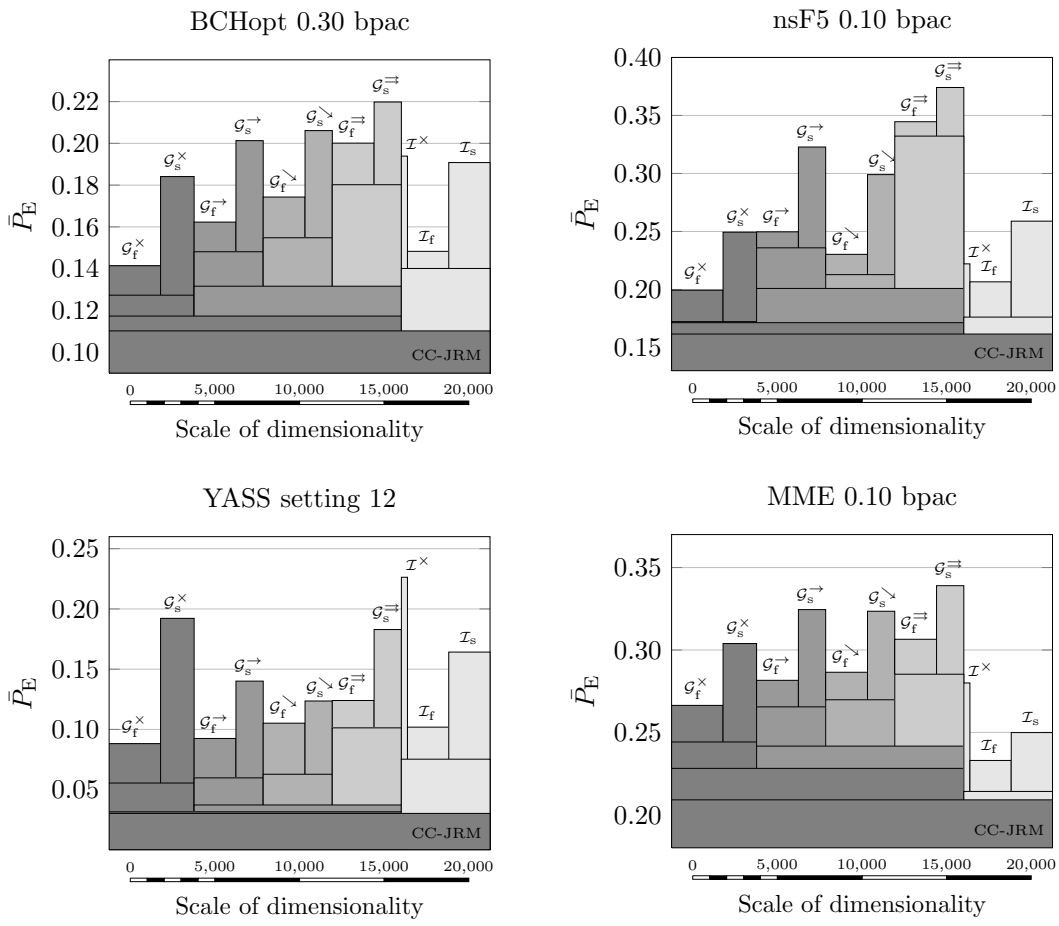


Figure 6.3.1: Systematic merging of the CC-JRM submodels and the progress of the testing error $\bar{P}_E$. See Section 6.3.1 for explanation of the graphs and their interpretation.

6.3 Investigative experiments

The purpose of this section is to study the contribution of the individual components of the CC-JRM to the overall performance. We also address the problem of finding a small subset of CC-JRM responsible for most of the detection accuracy for a fixed stego source. Our experiments are restricted only to selected steganographic methods and payloads. The notation follows Figure 6.1.1.

6.3.1 Systematic merging of submodels

In the first experiment, we consider the following disjoint and qualitatively different subsets of the CC-JRM: $\mathcal{G}_f^{\times}, \mathcal{G}_s^{\times}, \mathcal{G}_f^{\rightarrow}, \mathcal{G}_s^{\rightarrow}, \mathcal{G}_f^{\searrow}, \mathcal{G}_s^{\searrow}, \mathcal{G}_f^{\rightrightarrows}, \mathcal{G}_s^{\rightrightarrows}, \mathcal{I}^{\times}, \mathcal{I}_f, \mathcal{I}_s$, and use them for steganalysis separately. Afterwards, we gradually and systematically merge them together, following the logic of Figure 6.1.1, until all of them are merged into the CC-JRM. All considered feature sets are Cartesian calibrated, yielding double the dimensionalities shown in Figure 6.1.1. The experiment was performed on the following steganographic schemes: BCHopt 0.30 bpac, nsF5 0.10 bpac, YASS setting 12, and MME 0.10 bpac. The training procedure was identical to the one used in the experiments of Section 6.2: training on a randomly selected half of the CAMERA database, testing on the other half. The obtained performance is reported in Figure 6.3.1 in terms of $\bar{P}_E$.

In Figure 6.3.1, every submodel is represented by a bar whose height is the $\bar{P}_{\mathrm{E}}$. Conveniently, the width of each bar is proportional to the dimensionality of the corresponding submodel, allowing thus a continuous perception of the feature space sizes. For example, the rather thin bar of $\mathcal{I}^{\times}$ can be immediately perceived as more than five times smaller than the neighboring $\mathcal{I}_{\mathrm{f}}$. Intuitively, the union of several submodels is represented by an overlapping bar whose width is equal to the sum of its components. The overlapping bars do not interfere with the performance of their submodels because merging always decreases the error. For example, see the performance of submodels $\mathcal{G}_{\mathrm{f}}^{\rightrightarrows}$ and $\mathcal{G}_{\mathrm{s}}^{\rightrightarrows}$ in the top left graph (BCHopt). Their individual errors $\bar{P}_{\mathrm{E}}$ are 0.20 and 0.22, respectively, and their union (denoted $\mathcal{G}^{\rightrightarrows}$ in Figure 6.1.1) yields error 0.18, thence the corresponding height of the lower, wider bar. After adding additional submodels $\mathcal{G}_{\mathrm{f}}^{\rightarrow}, \mathcal{G}_{\mathrm{s}}^{\rightarrow}, \mathcal{G}_{\mathrm{f}}^{\searrow}, \mathcal{G}_{\mathrm{s}}^{\searrow}$, the error can be seen to drop further to roughly 0.13. The final performance of the CC-JRM is always represented by the lowest bar spanning the whole width of the graph. Finally, the readability is further improved by using different shades of gray for different types of submodels.

Figure 6.3.1 reveals interesting information about the types of features that are effective for attacking various steganographic algorithms. The four selected steganographic methods represent very different embedding paradigms, which is why the individual submodels contribute differently to the detection. The contribution of the integral features $\mathcal{I} = \{\mathcal{I}^{\times}, \mathcal{I}_{\mathrm{f}}, \mathcal{I}_{\mathrm{s}}\}$, for example, seems to be rather negligible for YASS because steganalysis *without* $\mathcal{I}$ delivers basically the same performance. For the other three algorithms, however, integral features noticeably improve the performance. This is most apparent for MME where the integral features $\mathcal{I}$ perform better than the rest of the features together, despite their significantly lower dimensionality. As another example, compare the individual performance of the DCT-mode specific features extracted directly from absolute values of DCT coefficients, $\mathcal{G}^{\times} = \{\mathcal{G}_{\mathrm{f}}^{\times}, \mathcal{G}_{\mathrm{s}}^{\times}\}$, with the DCT-mode specific features extracted from the differences, $\mathcal{G}_{\mathrm{diff}} = \{\mathcal{G}_{\mathrm{f}}^{\rightarrow}, \mathcal{G}_{\mathrm{s}}^{\rightarrow}, \mathcal{G}_{\mathrm{f}}^{\searrow}, \mathcal{G}_{\mathrm{s}}^{\searrow}, \mathcal{G}_{\mathrm{f}}^{\rightrightarrows}, \mathcal{G}_{\mathrm{s}}^{\rightrightarrows}\}$. While for nsF5, $\mathcal{G}_{\mathrm{diff}}$ does not improve the performance of $\mathcal{G}^{\times}$ much, both seem to be important for the other three algorithms and especially for YASS.

We conclude that there is no subset of CC-JRM that is universally responsible for majority of detection accuracy across different steganographic schemes. The power of CC-JRM is in the *union* of its systematically built components, carefully designed to capture different types of statistical dependencies.

6.3.2 Forward feature selection

Despite its high dimension (22,510), ensemble classifiers make the training in the CC-JRM feature space computationally feasible. In fact, the bottleneck of steganalysis now becomes the feature extraction rather than the actual training of the classifier. To give the reader a better idea, we measured the running time needed for steganalysis of nsF5 at 0.10 bpac using the CC-JRM. The extraction of features from $6,500$ images took roughly 18 hours, while, on the same machine,[3] the classifier training took on average 5 minutes. From the practical point of view, the *testing* time may be an important factor – after the classifier is trained, the time needed to make decisions should be minimized. Although projecting the CC-JRM feature vector of the image under inspection into eigen-directions of individual FLDs of the ensemble classifier consists of a series of fast matrix multiplications, the extraction of the 22,510 complex features is quite costly. Therefore, one may want to consider investing more time into the training procedure, and perform a supervised feature selection in order to reduce the number of features needed to be extracted during testing, while keeping satisfactory performance. Note that we are interested specifically in feature selection rather than general dimensionality-reduction as the goal is to minimize the number of features needed.

Unfortunately, as shown in the previous investigative experiment (Figure 6.3.1), there is no compact subset of CC-JRM that would be universally effective against different types of embedding modifications. However, if we *fix* the steganographic channel, the problem becomes feasible. This situation is analogical to the spatial domain where we studied several feature selection strategies, also for a

[3] Dell PowerEdge R710 with 12 cores and 48GB RAM when executed as a single process.

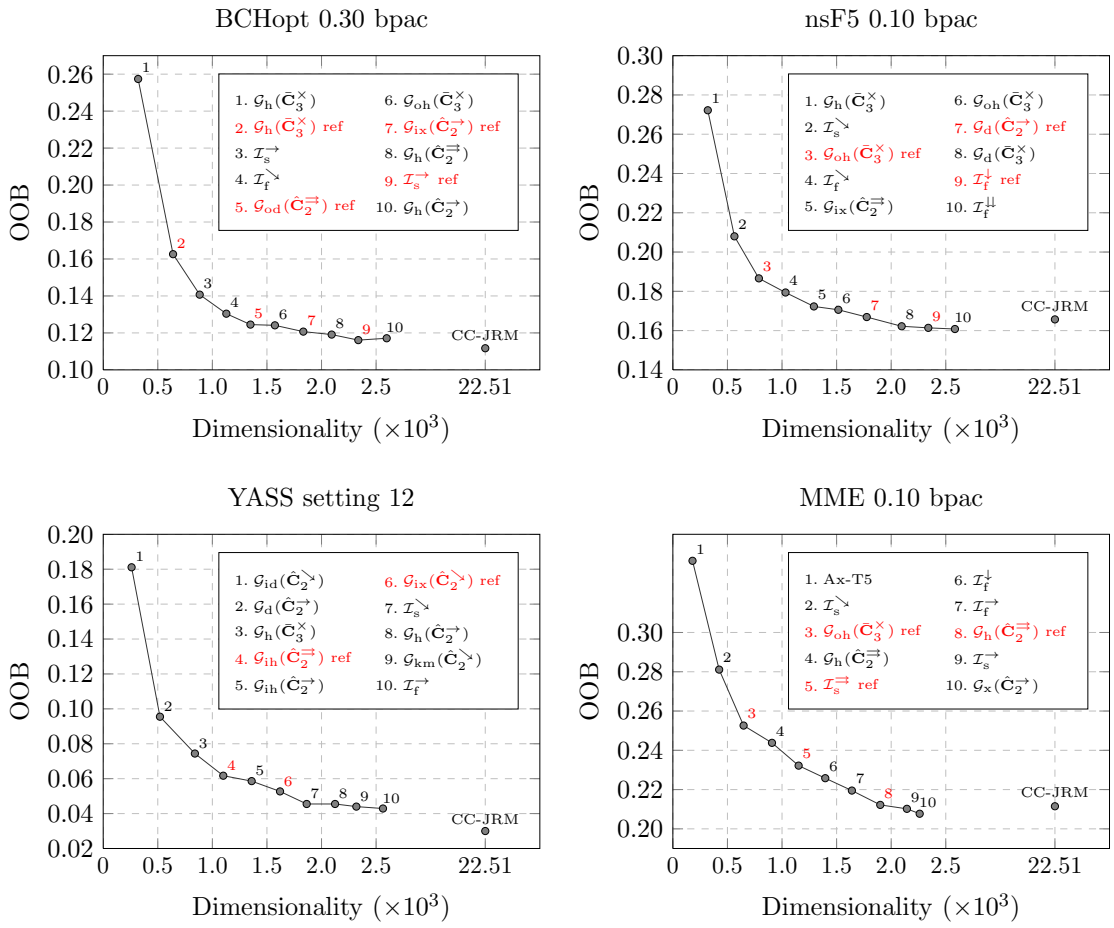


Figure 6.3.2: The result of the ITERATIVE-BEST feature selection strategy applied to four qualitatively different steganographic methods. The reported values are out-of-bag (OOB) error estimates calculated on the training set (half of the CAMERA database). For reference, we include the OOB error of the full CC-JRM feature space.

fixed steganographic channel (see Chapter 5.2.2). We will take use of the results from the spatial domain and, in order to demonstrate the feasibility of feature selection here in JPEG domain, apply the ITERATIVE-BEST strategy, the best-performing feature selection technique from the spatial domain (see Figure 5.2.3), to the individual submodels of CC-JRM.

We consider $N = 2 \times 51 = 102$ submodels of the CC-JRM, treating the submodels and their reference submodels coming from Cartesian calibration separately, and greedily minimize the OOB error through an iterative process. In particular, once $k \geq 0$ submodels are selected, we add the one that leads to the biggest drop in the OOB error estimate when all $k + 1$ submodels are used as a feature space. The procedure is stopped after 10 iterations.

We performed this feature selection strategy on BCHopt 0.30 bpac, nsF5 0.10 bpac, YASS setting 12, and MME 0.10. The results are shown in Figure 6.3.2. The individual graphs show the progress of OOB error estimates for $k \leq 10$ as well as the list of the selected submodels. We follow the notation of the submodels introduced in Figure 6.1.1 and distinguish the reference-version of the submodels by red color and adding the suffix "ref." For comparison, we also include the OOB-performance when the entire CC-JRM is used.

Figure 6.3.2 clearly demonstrates that it is indeed possible to obtain performance similar to CC-JRM

with as few as one tenth of its submodels, reducing thus the testing time by one order of magnitude. The selected submodels are generally different and algorithm-specific, which confirms our claim that there is no universally effective subset of CC-JRM.

The results provide us with another very interesting insight. Quite surprisingly, the appearance of a *reference* submodel often does not imply that the original version of the same submodel has been previously selected. In other words, a reference submodel may be useful as a complementary feature set to *other* types of features as well. Note, for example, the fourth selected submodel for YASS or the third for nsF5 and MME. This phenomenon indicates the intrinsic complexity of relationships among all extracted features and their reference values, and supports the hypothesis that appeared in [77]: individual features of complex feature spaces serve *each other* as references. For high-dimensional spaces, the concept of Cartesian calibration can thus be viewed simply as model enrichment that makes the feature space more diverse.

6.4 Summary

In this chapter, we constructed a 22,510-dimensional rich model for steganalysis of JPEG images, called CC-JRM. Similarly to the spatial domain rich model (SRM) introduced in Chapter 5, the CC-JRM consists of many simple submodels, each capturing a different type of dependencies among image coefficients. Unlike in the spatial domain, however, here we utilized two different types of dependencies, spatial (inter-block) and frequency (intra-block) dependencies among DCT coefficients of JPEG images. Furthermore, we incorporated the concept of Cartesian calibration, discussed in Chapter 4.2 of this dissertation. Matlab implementation of the CC-JRM, as well as the other feature sets used in this chapter, is available at `http://dde.binghamton.edu/download/feature_extractors`.

The novelty of the proposed CC-JRM w.r.t. our previously proposed rich models for steganalysis of JPEG images [78, 81] is at least three-fold: 1) we view the absolute values of DCT coefficients in a JPEG image as 64 weakly dependent parallel channels and separate the joint statistics by individual DCT modes; 2) to increase the model diversity, we form the same model from *differences* between absolute values of the DCT coefficients; 3) we add integral joint statistics between coefficients from a wider range of values to cover the case when steganographic embedding largely avoids disturbing the first two models. Finally, the joint statistics are symmetrized to compactify the model and to increase its statistical robustness. We would like to point out that the proposed approach necessitates usage of scalable machine learning, such as the ensemble classifier described in Chapter 3.

We showed that CC-JRM outperforms all previously published models of JPEG images (for steganalysis purposes) across six modern steganographic algorithms and a wide range of payloads. Furthermore, we showed that the union of rich models extracted from both domains, i.e., the merger of CC-JRM and SRM denoted as J+SRM in Table 6.1, further improves steganalysis across all six tested stego schemes and all payloads. This confirms that steganalysis benefits from multiple-domain approaches, and suggests that additional improvement may be achieved by adding even more transform domains, e.g., the wavelet domain.

The investigative experiments from Section 6.3 indicate that the proposed CC-JRM does not contain any universally effective subset that could replace CC-JRM while keeping its performance across different stegoschemes. However, if we are to construct a targeted steganalyzer for detection of a selected steganographic method, it is possible to significantly reduce the dimensionality by supervised feature selection. These conclusions are analogical to the ones observed in the spatial domain.

The last experiment of Section 6.3 showed that reference features are often useful even without their original feature values, which sheds more light on the real benefit of Cartesian calibration in high dimensions.

Chapter 7

Conclusion

This PhD dissertation is a comprehensive and a self-contained exposition of a novel framework for steganalysis of digital images based on rich image representations and ensemble classification. Parts of this work were published as full-length papers at various journals and conferences, including the ACM Multimedia Security Workshop, Information Hiding Workshop, and SPIE Electronic Imaging.

The key concept of the proposed framework is the ensemble classifier, a scalable machine-learning alternative to the complex SVMs. The ensemble classifier is built by fusing decisions of simple base learners constructed on random subspaces of the original feature space. It offers comparable accuracy to SVMs at a fraction of the computational cost, and scales more favorably w.r.t. both the dimensionality of the feature space and the number of training images.

Replacing SVMs with ensemble classifiers enables us to approach the feature-space building process in a systematic and a clean way. An important property of successful feature spaces is *diversity*, and therefore we build the model as a collection of a large number of smaller and simpler submodels, each of them capturing different type of dependencies among image coefficients. As a result, we obtain a high-dimensional rich statistical descriptor of images, the so-called rich model.

To demonstrate the power of the proposed methodology, we construct rich models for the two most commonly used image formats, the raster format and the JPEG. Our Spatial domain Rich Model (SRM) is a $34,671$-dimensional feature space consisting of co-occurrences of neighboring samples of noise residuals obtained by a range of linear and non-linear filters. The Cartesian-calibrated JPEG domain Rich Model (CC-JRM) is a $22,510$-dimensional space consisting of submodels capturing different types of frequency- and spatial-dependencies among DCT coefficients of JPEG images. Both rich models, combined with the ensemble classifier, are shown to substantially outperform previous art across a wide range of steganographic schemes hiding data in both domains. We note that the implementation of the ensemble classifier, as well as the codes for extraction of both rich models, are available for download at `http://dde.binghamton.edu/download`.

A possible drawback of using high-dimensional models is an increased complexity of feature extraction which may be a limiting factor for online applications. Furthermore, the improved performance of steganalysis on *a given cover source* may decrease its robustness to the cover-source mismatch, i.e., when the testing images come from a different source than the classifier is trained on. In fact, we observed this performance degradation during the steganalysis competition BOSS – when the ensemble classifier was trained on images coming from the Leica M9, the performance dropped on images coming from Panasonic Lumix DMC-FZ50.

Steganalysis using rich models is a novel framework that offers several promising directions for future research. Here we list a few topics that appear worth exploring:

- Using rich models for construction of more secure steganographic schemes. Minimizing a carefully designed distortion function (in the rich model) and combining it with an appropriate coding [36] seems to be a promising way of doing so.

- Applying rich image representations to other digital image forensics tasks, such as problems dealing with media integrity (forgery detection), authentication, or processing history recovery.
- Developing a similar framework for other media types, including audio and video signals.
- Extension of the rich-model-based ensemble classifier into a quantitative steganalyzer. One of the possible approaches has been proposed in [112].

Appendix A

Steganographic methods

A.1 Jsteg

Jsteg is the first steganographic algorithm for JPEG images developed in 1997 by Korejwa and later improved by Upham [129]. Jsteg embeds messages by replacing the Least Significant Bits (LSBs) of the quantized DCT coefficients with message bits. Jsteg does not embed in coefficients equal to 0 or 1 because too many non-zero coefficients would appear in higher frequencies, which would lead to perceptible and statistically detectable artifacts.

The first version of Jsteg embedded individual secret data bits sequentially, which turned out to be accurately detectable by the histogram attack [133]. An improved version of Jsteg embeds data along a pseudo-random path generated from a secret stego key. Only this randomized version of Jsteg is considered in this dissertation (in the experiments of Chapter 4) and is referred to as Jsteg. Due to its low security, Jsteg is not considered in later Chapters.

We note that the security of Jsteg could be markedly improved by decreasing the number of embedding changes by incorporating a coding framework. For example, the near-optimal coding scheme proposed in [36] could be used. This could also elegantly resolve Jsteg's security flaw – the disturbance of histogram symmetry discussed in Chapter 4.1.

A.2 OutGuess

OutGuess is a JPEG domain steganographic technique developed by Niels Provos [113] as a response to Westfeld's statistical chi-square attack [133]. OutGuess fully preserves the histogram of DCT coefficients in the image and thus cannot be detected using first-order statistics.

OutGuess is an example of the so-called statistical restoration. It embeds data in two subsequent passes. In the first pass, a pre-defined portion of the DCT plane is used for embedding. In the second pass, the previously unused coefficients are utilized to fully restore the original histogram of DCT coefficients. Similar to Jsteg, OutGuess skips DCT coefficients equal to 0 or 1 and uses LSB flipping as the embedding operation. The implementation of OutGuess is available at `http://www.outguess.org`.

The histogram of DCT coefficients is a poor model of JPEG images, and even though OutGuess cannot be detected using first-order statistics, it has been successfully steganalyzed using higher-order statistics numerous times [43, 45, 135, 71, 22].

A.3 JPHide&Seek (JPHS)

JPHS is a JPEG domain steganographic algorithm developed in 1998 by Allan Latham. Its implementation is publicly available at http://linux01.gwdg.de/~alatham/stego.html, and we used it in the experiments of Chapter 4. Similarly as Jsteg, JPHS was dropped from our steganalysis experiments in Chapter 6, since its security is rather low.

Even though the mechanism of JPHS has not been published, a careful inspection of cover images and embedded stego images reveals a few hints about its embedding process:

- For positive DCT coefficients, the embedding operation is a simple LSB flipping, while for negative coefficients it is "shifted" LSB flipping. Consequently, the flipping is symmetrical w.r.t. zero and thus the embedding does not disturb histogram symmetry.
- JPHS embeds also into DC terms. In fact, for smaller payloads, *only* DC terms are used. Steganalyzers ignoring DC terms may thus mistakenly consider JPHS embedding at lower rates as a secure algorithm.
- Even when a short message is embedded, the amount of distortion is rather large, most likely due to the large message header. In fact, according to our limited experiments on the CAMERA database, even the stego images with zero message (only the message header is present) could be distinguished from cover images using CC-PEV features with $P_{\mathrm{E}} \sim 10\%$. We postpone the details about this simple experiment to Appendix C.2.

A.4 StegHide

StegHide [57] represents a graph-theoretic approach to steganography, and its code is available at http://steghide.sourceforge.net/. Similarly to JPHS, StegHide also uses the DC terms for data hiding, and embeds quite a large message header, which makes it detectable even when a zero-length message is communicated (see Appendix C.2).

Steghide was steganalyzed in Chapter 4, and since its detection error at a rather small payload 0.05 bpac was already $\sim 1\%$ using CC-PEV features, we did not attack this algorithm in Chapter 6.

A.5 F5 (nsF5)

The F5 algorithm is due Andreas Westfeld [132] and played an important role in the development of modern steganographic schemes. It introduced a novel design element, the so-called *matrix embedding*, a coding technique that substantially decreases the number of embedding changes, especially for smaller payloads, and thus allows Alice to communicate larger messages with the same amount of distortion.

Another novel element of F5 is its embedding operation – instead of LSB flipping, the absolute value of the to-be-changed DCT coefficients is always decreased, and thus the overall histogram shape is preserved after embedding. F5 does not skip coefficients that are equal to –1 or 1. However, if these need to be modified to zero, the same message bit is re-embedded on the next coefficient of the pseudo-random path over the image, as the recipient extracts the message only from the non-zero coefficients. This phenomenon is called *shrinkage*.

The nsF5 (non-shrinkage F5) algorithm [51] is an improved version of F5, where the problem of shrinkage is eliminated by using a more sophisticated coding scheme [47, 36] rather than by re-embedding. This way, the same payload can be communicated using fewer embedding changes, yielding higher security. Since the DCT coefficients equal to -1 or 1 are the most common non-zero coefficients in JPEG images, the improvement of nsF5 over F5 is substantial (see, for example, [51]).

For experiments in this dissertation, we only used the non-shrinkage version of F5. In particular, we used the simulator of nsF5 that makes the embedding changes as if an optimal binary matrix coding scheme was used. Its Matlab implementation is available at http://dde.binghamton.edu/download/nsf5simulator/. We note that a near-optimal practical implementation can be achieved using syndrome-trellis codes [36].

A.6 MBS

Model-based steganography (MBS) is a JPEG-domain algorithm due Phil Sallee [116]. It attempts to preserve a *model* of DCT coefficients, created separately for each AC DCT mode (DC terms are omitted). In particular, for every AC mode, the algorithm fits a generalized Cauchy distribution whose parameters are estimated using the maximum-likelihood procedure using the sums of the histogram bins forming individual LSB pairs as embedding invariants. Once the model is constructed, the portions of the uniformly distributed message bitstream are pre-biased using an entropy decompressor to match the conditional probabilities extracted from the model.

The second version of the MBS algorithm, commonly referred to as MBS2, was proposed later in [117]. It follows the paradigm of statistical restoration and introduces additional changes in the second pass of the embedding process in order to preserve the so-called blockiness, a measure of spatial discontinuities along the boundaries of 8×8 blocks of DCT coefficients. This enhancement is called *deblocking*. However, the additional deblocking changes have been shown to make the algorithm more detectable [123, 107], and thus we do not consider MBS2 in this dissertation. Instead, we steganalyze only the original MBS, sometimes also called MBS1.

For all experiments in this PhD thesis, we used the implementation of MBS available at http://www.philsallee.com/mbsteg.

A.7 MME

MME, a JPEG domain algorithm proposed by Kim *et al.* [72], stands for Modified Matrix Encoding. It employs matrix embedding and uses a side information in the form of the decompressed image at sender to minimize an appropriately defined distortion function. It considers making more changes than one, provided their combined distortion is lower. The algorithm is commonly denoted as MMEx, where the symbol $x \in \{2, 3, \ldots\}$ stands for the maximal number of changes considered. For example, MME3 combinatorially evaluates the distortion caused by flipping the coefficients individually, in all possible pairs, and in triples within each embedding block. Within the scope of this PhD thesis, we use solely MME3, and denote it shortly as MME.

The implementation of MME provided by authors is written in Java. As mentioned in the previous paragraph, MME is a side-informed algorithm that accepts images in the spatial format at the input.[1] The program outputs JPEG images of a desired quality. As such, it contains its own implementation of DCT written in Java (Java JPEG encoder). This seemingly unimportant detail can significantly influence the outcome of steganalysis, if not taken into account, see Appendix C.1 for more details on this topic. At this point, we merely state that in order to correctly steganalyze the *impact* of embedding, cover images need to be created using the same JPEG encoder as stego images.[2]

Furthermore, the Java JPEG compressor used in MME inserts the following comment into the header of the compressed JPEG image: "JPEG Encoder Copyright 1998, James R. Weeks and BioElectroMech." This quite rare note, if present in a JPEG file, may by itself raise suspicion of the

[1] In fact, the Java implementation accepts JPEG images at the input as well. In that case, the image is first decompressed into the spatial domain. Starting with JPEG, however, leads to the significantly lower security as the full potential of the distortion function is not utilized.

[2] We note that similar issue exists with the original implementation of the F5 algorithm which decompresses the image prior embedding and then recompresses it using its own DCT implementation.

Setting	QF_{h}	DBs	B	rep	bpac
YASS 1	65,70,75	3,7	9	0	0.110
YASS 2	75	-	9	0	0.051
YASS 3	75	-	9	1	0.187
YASS 4	65,70,75	2,5	9	0	0.118
YASS 5	50,55,60,65,70	3,7,12,17	9	0	0.159
YASS 6	75	-	10	0	0.031
YASS 7	65,70,75	3,7	10	0	0.078
YASS 8	75	-	10	1	0.138
YASS 9	65 70 75	3,7	9	2	0.237
YASS 10	75	-	10	2	0.159
YASS 11	75	-	11	1	0.114
YASS 12	65 70 75	3,7	11	0	0.077

Table A.1: Twelve settings for YASS as introduced in [82]. The explanation of the columns is in the text.

warden who observes the communication channel and thus serve as a detector of stego communication. This is an example of the so-called *system attack*, which highlights the importance of every implementation decision for the practical security of the steganographic tool.

A.8 YASS

YASS (Yet Another Steganographic Scheme) hides data robustly in a transform domain. In particular, it uses a quantization-index-modulation type of embedding in randomly positioned 8×8 blocks of the image. After the embedding finishes, the individual blocks are decompressed back to the spatial domain, and the whole image is re-compressed one more time into JPEG. The algorithm was originally proposed by Solanki *et al.* [126] and later improved by Sarkar *et al.* [118].

In [82], we steganalyzed YASS, and introduced 12 different parameter settings influencing its payload, distortion, and detectability. To make this PhD dissertation self-contained, all 12 settings are listed in Table A.1. The brief explanation of the individual columns of the table is as follows. The column 'QF_{h}' contains the hiding quality factor(s), while 'B' stands for the size of the big blocks withing which the positions of 8×8 hiding blocks are randomly generated. Settings 1, 4, 5, 7, 9, and 12 incorporate the mixture of hiding quality factors QF_{h} based on block variance, the improvement of YASS proposed in [118]. For example, for YASS 1 (follow the setting of YASS 1 in Table A.1): if the block variance is in the interval [0,3), $QF_{\mathrm{h}} = 65$ is used for hiding. If the block variance is in the interval [3, 7), $QF_{\mathrm{h}} = 70$ is used. Finally, if the block variance is ≥ 7, $QF_{\mathrm{h}} = 75$ is used. The decision boundaries (DBs) for these different decisions are shown in the column 'DBs'.[3]

Settings 3, 8, 9, 10, and 11 use the second improvement proposed in [118], the so-called attack-aware iterative embedding (the column '*rep*' is the number of iterations). Settings 2 and 6 do not use any extensions and correspond to the original version of YASS. The last column, 'bpac', is the average payload in bits per non-zero AC DCT coefficient computed across all images in the CAMERA database as YASS can embed only the full payload. The method for determining the payload is explained in details in [82]. In all the experiments in this thesis, the input cover images were always in the uncompressed format and the advertising quality factor of stego images was fixed to $QF_{\mathrm{a}} = 75$. With these choices, YASS appears to be the least detectable [75]. Also, all settings used the default choice of 19 AC DCT coefficients for embedding.

[3] From our experiments, it appears that the actual choice of DBs does not influence the results much.

Even though YASS is an easily detectable algorithm today [90, 82, 92, 79], it played an important role to clarify the real purpose of the process of feature calibration [77], see Chapter 4.2. In this dissertation, YASS is steganalyzed in Chapters 3.5, 3.6, 4.2, and 6, and the notation always follows Table A.1.

A.9 MOD

MOD (Model Optimized Distortion) is a relatively new steganographic algorithm for JPEG images proposed by Filler *et al.* [35]. It follows the principle of a minimum-impact embedding. In particular, it attempts to minimize a distortion function whose parameters are *learned* by minimizing the margin of a linear SVM on a small sample of cover and stego features in a chosen feature space.

In [35], the MOD algorithm used the 548-dimensional CC-PEV feature space as a model. To show that the distortion was not overtrained to this model, the authors steganalyzed MOD with the CC-PEV set with a slightly different cropping in calibration (see Chapter 4.2) as well as with the Cross-Domain Feature set (CDF) obtained by merging CC-PEV and the 686-dimensional SPAM vector [104] computed from images represented in the spatial-domain. No signs of overtraining were revealed and MOD was reported to be significantly more secure than the nsF5 algorithm.

In Chapter 4.3 of this dissertation, however, we reveal that MOD is, in fact, highly detectable when using an appropriately enlarged cover model of a relatively low dimension.[4] Therefore, we conclude that the MOD steganography is overtrained to an incomplete model, similarly as FCM [76].

In spite of the low security of MOD, we see the merit of the work [35] in the *methodology* rather than in its specific realization in terms of the insecure MOD algorithm because one can easily replace the CC-PEV image model with a more complete statistical descriptor of images. For example, an improved version of the MOD algorithm could be obtained by replacing CC-PEV with the JPEG domain rich model described in Chapter 6 of this thesis. It is not clear, however, how to parametrize the distortion function or how well would such a technique resist rich-model-based steganalysis as it is increasingly more difficult to preserve a feature vector when its complexity and dimensionality increases.

A.10 BCH, BCHopt

BCH and BCHopt [115] are side-informed algorithms that employ BCH codes to minimize the embedding distortion in the DCT domain defined using the knowledge of unrounded DCT coefficients. BCHopt is an improved version of BCH that contains a heuristic optimization and hides message bits also into zeros. According to the experiments in [115], BCHopt is the most secure practical JPEG steganographic scheme up to date. This is confirmed in Chapter 6.2 of this dissertation, where BCHopt performs clearly the best of all tested stego methods.

Similarly as MME, BCH-based steganography uses side information in terms of the uncompressed image (from which the unrounded DCT coefficients are obtained). As such, its practical implementation needs to include a JPEG compressor and steganalysis needs to be performed with care. To be more specific, the same JPEG compressor needs to be used for creating cover images, otherwise the detection error is artificially decreased by detecting not only the embedding impact, but also differences in the JPEG compressor. We refer the reader to Appendix C.1 for more details on this issue.

[4] Another successful attack on MOD algorithm appeared in [92].

A.11 LSB replacement

Arguably, LSB (Least Significant Bit) replacement, sometimes called LSB embedding, is the simplest possible embedding mechanism – the LSBs of individual coefficients are replaced with message bits. Due to its simplicity, straightforward implementation, and high embedding capacity, many existing steganographic programs employ some version of LSB replacement, in both spatial and JPEG domains.

Steganalysis of LSB embedding in spatial domain is a well developed research area, and the asymmetric nature of its embedding operation (even values are increased while odd values are decreased) gave rise to many targeted structural attacks, such as, RS analysis [42], Sample Pairs Analysis (SPA) [28, 65], least-squares steganalysis [67, 66], and the predictor-based weighted stego-image analysis (WS) [41, 68].

We do not steganalyze LSB replacement in this dissertation. Instead, we perform steganalysis of its modification, the ±1 embedding, which is described next.

A.12 LSB matching (±1 embedding)

±1 embedding, also called LSB matching, is a trivial modification of LSB replacement. It also embeds message bits as LSBs of visited coefficients, but their value is randomly increased or decreased (except the values 0 and 255 which are *only* increased or decreased, respectively). This simple modification disables structural attacks.

±1 embedding is a non-adaptive data hiding scheme whose security could be enhanced by incorporating matrix embedding. Within the scope of this dissertation, we assume that the algorithm is implemented with ternary matrix embedding that is optimally coded to minimize the number of embedding modifications. In particular, the relative payload α bpp (bits per pixel) can be embedded with change rate $H_3^{-1}(\alpha)$, where $H_3^{-1}(x)$ is the inverse of the ternary entropy function $H_3(x) = -x\log_2 x - (1-x)\log_2(1-x) + x$. For more details, see, e.g., Chapter 8 in [40].

In the experiments of Chapter 5, the embedding impact of ±1 embedding with change rate α bpp was simulated by randomly flipping $H^{-1}(\alpha)$ pixels in the image.

A.13 HUGO

HUGO (Highly Undetectable steGO) is an adaptive spatial-domain steganographic algorithm proposed by Pevný *et al.* [105]. It follows the paradigm of the so-called minimum-embedding-impact steganography that embeds a given payload while minimizing the impact of introduced modifications in terms of a suitably defined distortion function. Such a formulation then formally becomes source coding with a fidelity criterion for which near-optimal coding schemes were developed [36, 32].

More details on the embedding mechanism of HUGO, including the formal definition of its distortion function, appear in Chapter 4.3.3 of this dissertation where we reveal a security flaw of HUGO consisting in the abrupt end of its high-dimensional model. This weakness can be eliminated by increasing the threshold value to $T = 255$ (see Chapter 4.3.3 for more details).

For the steganalysis experiments in Chapter 5, we used the HUGO embedding simulator available from the BOSS website, `http://www.agents.cz/boss`, with $\sigma = 1$ and $\gamma = 1$ for the parameters of the distortion function (4.3.4) and the switch –T 255.

A.14 Edge-Adaptive (EA) algorithm

Edge-Adaptive (EA) algorithm was proposed by Luo *et al.* [94]. It is a spatial-domain adaptive scheme that confines the embedding changes to pixel pairs whose difference in absolute value is larger than a certain threshold whose value is dynamically determined based on the payload. This confines the embedding changes to the regions around edges which are difficult to model statistically.

In spite of its adaptivity, EA algorithm is significantly less secure than HUGO, as showed in Chapter 5.3 of this dissertation. In fact, for larger payloads (0.4 bpp), the security of EA seems to be similar to the simple non-adaptive ± 1 embedding.

Appendix B

Image Datasets

It is known that the properties of images used for steganalysis experiments, e.g., the image sizes or their JPEG quality factor, significantly influence obtained detection results [75]. Therefore, all steganalysis experiments should be accompanied by a detailed description of the used image datasets as well as their complete pedigree.

Within the scope of this PhD dissertation, we used the following databases of digital images:

1. CAMERA. Internal database of our research group consisting of $6,500$ images originally acquired in their RAW format taken by 22 digital cameras spanning five camera brands. All images were converted to 8-bit grayscale, and resized using bilinear interpolation so that the smaller side of the image was 512 pixels (aspect ratio preserved). For experiments in the JPEG domain, all images were compressed with the JPEG quality factor 75 using Matlab's command `imwrite`.

2. BOSSbase (v0.92). The collection of $9,074$ images used during the BOSS (Break Our Steganographic System) competition [9], publicly available at the organizers' website, `http://www.agents.cz/boss`. The images were taken with seven digital cameras in their RAW format, converted to grayscale, and resized/cropped to 512×512 using the script provided by the BOSS organizers.

3. BOWS2. This database contains $10,000$ grayscale images of sizes 512×512 used in the BOWS2 watermarking competition [10]. These images are publicly available at the competition website, and we used them in Chapter 3.7.

Appendix C

Practical Considerations

The two most important elements of a feature-based steganalyzer are the constructed feature space $\mathcal{F}$ and the chosen classification tool. Once these are fixed, the experimental procedure seems to be straightforward: generate stego images, extract cover- and stego-features, create training and testing sets, train the classifier, and finally evaluate the security of a given steganographic scheme on the testing set.[1] This procedure, however, contains several pitfalls that may undesirably influence the results. For completeness of this dissertation, this appendix covers the following areas researchers need to be aware of:

- Troubles with JPEG compressor
- Message header of steganographic schemes
- Cover-stego pairs and their implications for classification

C.1 JPEG compressor

Steganalysis feature spaces are designed to be sensitive to very subtle embedding impacts. For example, the algorithm nsF5, described in Appendix A.5, changes on average 3.12% DCT coefficients in a given JPEG image when embedding a message of a relative payload 0.20 bpac. In spite of this small distortion (less than 1 out of 30 nonzero coefficients is increased or decreased by 1), modern steganalysis is capable of detecting nsF5 stego images at this payload with error smaller than 5%, see Figure 6.2.1 in Chapter 6.

Therefore, it is not surprising that features originally developed for steganalysis have been found useful for other forensics tasks. For example, in [73], authors used SPAM features for detection of median-filtered images. Or in [109] and [91], authors used feature-based approach to identify double-compressed images and cropped images, respectively.

Here, we ask whether it is possible to use steganalysis features for distinguishing between two different *implementations* of the Discrete Cosine Transform, the core component of the JPEG compression. A simple investigative experiment described in Section C.1.1 shows that certain JPEG compressors are, indeed, distinguishable very reliably. We discuss the implications of this finding for steganalysis in Section C.1.2.

[1] These steps are described in full details in Chapter 2.6.

	IMWRITE	FFT	CONVERT	JAVA	BATCH	XNVIEW
IMWRITE	×	.2937	.2929	.2944	.2084	.0136
FFT		×	.4184	.4593	.2731	.0135
CONVERT			×	.4470	.2720	.0147
JAVA				×	.2743	.0138
BATCH					×	.0186
XNVIEW						×

Table C.1: Detection of six different JPEG compressors using the ensemble classifier with CC-PEV features. The reported values are means of the testing errors P_{E} obtained from 10 independent database splits.

C.1.1 Detecting JPEG compressors

We consider the following implementations of JPEG compression:

1. IMWRITE – Matlab's `imwrite` command; uses C library `wjpg8c`,
2. FFT – Matlab's `fft` command; uses C library `fftw`,
3. CONVERT – ImageMagick's `convert`; free command-line tool,
4. JAVA – Java JPEG encoder,[2]
5. BATCH – BatchPNGtoJPG; free software for Windows OS,
6. XNVIEW – XnView; free image processing tool; fast option enabled.

The first compressor is a simple call of Matlab's `imwrite` function. Due to its ease of use, IMWRITE is the JPEG compressor used almost exclusively within the scope of this dissertation. We note that the C library `wjpg8c` used by IMWRITE seems to be a popular JPEG compressor – it is used in other image processing tools, for example in IrfanView or in Phil Sallee's Matlab JPEG Toolbox[3]. Another Matlab option for transforming spatial domain images into DCT coefficients is to perform the compression manually, block-by-block, using the `fft` function – the second compressor in the list above. Compressor 3, CONVERT, is part of ImageMagick,[4] a linux-friendly command-line image processing tool. The fourth compressor is a Java-based encoder that is used in the implementation of MME. It was also used in the original version of F5 algorithm. The last two compressors, BATCH and XNVIEW, are publicly available freeware tools enabling batch JPEG compression of a given set of images. In case of XNVIEW, we used the "fast" option in the dialog for JPEG compression.

All images from our CAMERA database were taken in their native resolution, resized to make the smaller side 512 pixels with aspect ratio preserved, and JPEG compressed with the quality factor 75 using all six compressors. We verified that all resulting JPEG images had identical quantization tables. Then we used the CC-PEV features [77] and trained the ensemble classifier (Chapter 3) to distinguish between each pair of JPEG compressors. The resulting detection errors P_{E} (means over 10 different splits of the database into training and testing sets) are reported in Table C.1.

The obtained results reveal that XNVIEW uses a significantly distinct DCT implementation – it can be distinguished from all remaining compressors with an error rate below 2%. This could be accounted to the chosen "fast" method of JPEG compression. If the "slow" option was used, the

[2] JPEG Encoder Copyright ©1998, James R. Weeks and BioElectroMech.
[3] http://www.philsallee.com/jpegtbx
[4] http://www.imagemagick.org

Algorithm	IMWRITE	FFT	CONVERT	JAVA	BATCH	XNVIEW
BCHopt (0.10 bpac)	.4611	.3047	.3012	.3030	.2133	.0130
BCHopt (0.15 bpac)	.4119	.3058	.3041	.3074	.2181	.0124
MME (0.10 bpac)	.1557	.2571	.2551	.2621	.1947	.0113
MME (0.15 bpac)	.1007	.1711	.1695	.1725	.1356	.0097

Table C.2: Detection of BCHopt and MME at selected payloads using the ensemble classifier with CC-PEV features. Reported values are means of the testing errors P_{E} obtained from 10 independent database splits. Individual columns correspond to different JPEG compressors used for generating cover images. In each row, the compressor consistent with the one employed by the steganographic algorithm is highlighted.

resulting images are identical to the ones produced by IMWRITE. In other words, XNVIEW with the "slow" option uses C library `wjpg8c` as well.

Quite surprisingly, one could distinguish between the two different Matlab compressions (IMWRITE vs. FFT) with error rate around 30%. A closer inspection reveals that JPEG images produced by IMWRITE differ from images produced by FFT on average in about 1.2% DCT coefficients. This roughly corresponds to the number of changes created by nsF5 embedding at payload 0.10 bpac!

Finally, according to the results shown in Table C.1, it seems that JPEG compressors FFT, CONVERT, and JAVA are quite similar, as their mutual errors are always above 40%.

The differences in the resulting JPEG images of individual compressors may be caused by several factors – by different implementations of the DCT, different order and precision of arithmetic operations (and due to finite computer precision), and by source codes compiled at different computer architectures. We discovered, for example, that the absolute value of all DCT coefficients created by IMWRITE is always greater or equal to the absolute value of the corresponding DCT coefficients created by FFT, suggesting consistently biased coefficient values before rounding.

C.1.2 Implications for steganalysis

The simple investigative experiment presented in the previous section confirms that steganalysis features are capable of distinguishing between different JPEG compressors, at least to a certain degree. It is important to realize that this may create an undesirable bias in steganalysis experiments if not performed carefully. More specifically, care needs to be taken when steganalyzing side-informed JPEG algorithms which start with the spatial domain image and perform JPEG compression as part of their embedding process. This is the case of BCHopt and MME, for example, both briefly described in Appendix A. In order to study solely the effects of the *embedding impact*, cover images need to be created using the same JPEG compressor. Otherwise, it is easy to see that the obtained errors would be artificially decreased by detecting also the differences in the JPEG compressor.

This is confirmed by the following experiment, where we attack the steganographic algorithms BCHopt and MME. Cover images were successively created by all six compressors listed in the previous section. For simplicity, we used the ensemble classifier with CC-PEV features for steganalysis, and report the results in terms of the mean values of P_{E} over 10 CAMERA database splits. See Table C.2 for the results.

Clearly, the choice of the JPEG compressor for creating cover images plays an important role and the obtained error rates vary significantly. In case of BCHopt, for example, IMWRITE leads to errors above 40%, while using XNVIEW results in errors below 2%. Compressors FFT, CONVERT, and JAVA deliver values around 30%. A careful inspection of the BCHopt implementation, kindly provided by the authors, reveals that the program uses Matlab's IMWRITE JPEG compressor.

Therefore, the correct error rates are those where cover images were also created by the IMWRITE JPEG compressor, i.e., BCHopt algorithm could be considered quite secure. In all other scenarios, the lower error rates are due to the differences between JPEG compressors and have little to do with the embedding mechanism. This claim is supported by the comparison of the first row of Table C.1 with the first two rows of Table C.2, as these are apparently correlated.

Similar observations could be made for the MME algorithm. In this case, the implicit JPEG compressor is JAVA, therefore, in order to correctly interpret the steganalysis results, the cover images need to be generated using the JAVA compressor. As Table C.2 shows, using other compressors leads to artificially lower errors. Note that the error rates for the case of FFT and CONVERT are similar to JAVA. This is consistent with the conclusion made in the previous section – the compressors FFT, CONVERT, and JAVA are quite similar.

To conclude, it is of utmost importance to be aware of the compressor issue when performing steganalysis experiments. If the cover images were generated inconsistently with the stego images, the detection results may lead to misinterpretations of the algorithm's steganographic security.

C.2 Message header

A plausible solution for the above–discussed problem with JPEG compressors may be the following idea. Instead of generating cover images, let us use stego images with a zero message. For generating zero–message stego images, we will use the same code as for generating regular stego images, so the whole issue with mismatched JPEG compressors is implicitly resolved and the only effect the classifier will be detecting is the actual embedding impact due to non-zero messages.

However, this seemingly correct reasoning has the following problem. Many steganographic techniques need to communicate a certain number of bits *regardless* of the message length. These bits may contain information about the message length, coding parameters, etc., and are present even for very small (or zero) messages. While this so-called message header is usually negligibly small, there are steganographic techniques whose message header is surprisingly large. In these cases, treating zero–message stego images as covers would be obviously an incorrect decision.

To demonstrate the importance of this issue, we steganalyze the algorithms JPHS and StegHide, both briefly described in Appendix A. We chose these two algorithms because of their alarmingly large message headers. The steganalysis was performed similarly as in Appendix C.1 – we used the CC-PEV features and the ensemble classifier and report the results in terms of the average testing error P_{E} taken over 10 CAMERA database splits into equally–sized training and testing sets. In Figure C.2.1, we show the error rates for a wide range of relative payloads for both algorithms. Two scenarios are compared. In the first case, we use regular cover images for classification. In the second case, we generate zero–message stego images and treat them as covers.

The results clearly indicate that the message header is an important factor that can strongly influence the results of steganalysis experiments. In both cases, treating stego images with zero message length as covers leads to significantly higher error rates. This could lead to misinterpretations of the security of the schemes. For example, JPHS could be considered perfectly secure at payload 0.05 bpac, as the detector's performance at this message length is equivalent to random guessing. However, the truth is that JPHS stego images at payload 0.05 contain a strong message header which, by itself, could be distinguished from ordinary cover images with a low error around 10% (using CC-PEV features). This obviously contradicts its perfect security.

Similar conclusions could be made for the algorithm StegHide, where the ensemble classifier could identify its large message header with an error rate around 30%. One may think that this security flaw is not relevant to modern steganographic schemes anymore. Indeed, both JPHS and StegHide are quite old algorithms whose security has been compromised numerous times in the past. However, we encountered this very same artifact in Chapter 6.2, Figure 6.2.1, when steganalyzing the algorithm MME. There, as well, the error rate at zero payload was better than random guessing, and the reason

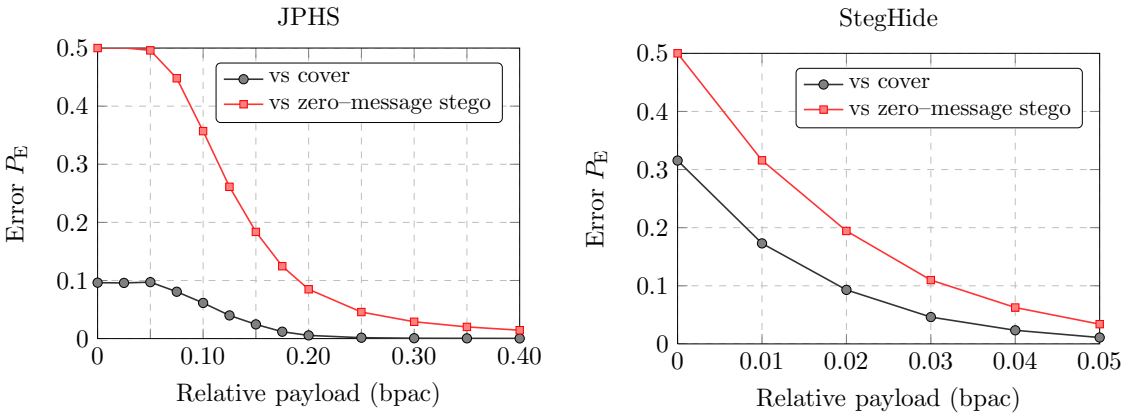


Figure C.2.1: Steganalysis of JPHS (left) and StegHide (right) using the ensemble classifier with CC-PEV features. We contrast the situation when regular cover images are used for classification with the case when zero–message stego images are treated as covers. The reported values of P_E are averages over 10 CAMERA database splits.

for this was the message header of MME inserted consistently into the upper left corner of every image.

C.3 Cover-stego pairs in steganalysis

Classification problems arising in steganalysis have a very specific nature – the individual training samples naturally form *pairs* of cover–stego features that originate from the same image. In the chosen feature space $\mathcal{F}$, these pairs are represented by two vectors with opposite labels lying close to each other. This is caused by the relatively small embedding distortions, compared to the variation of feature vectors over different image contents.

This intrinsic property of steganalysis feature spaces needs to be taken into account when performing experiments. First, when dividing the image database into training and testing parts, the cover–stego pairs (C-S pairs) need to be preserved. This is a commonplace precaution in the steganalysis research literature – obviously, if a cover image is used for classifier training, its stego-version cannot be used in the testing phase for evaluation of the system's security. However, the impact of C-S pairing if far more reaching than that and this very specific nature of classification problem needs to be constantly kept in mind.

To demonstrate the importance of C-S pairs in steganalysis, in this appendix we study its effects on cross-validation, a commonly used machine–learning procedure for optimizing classifier parameters. In particular, we show that the standard k-fold cross-validation, as implemented for example in LIBSVM [21], a widely used implementation of the support vector machine (SVM), is *not* suitable for steganalysis and may result in a suboptimal performance and a striking discrepancy between the predicted and the real testing error. Instead of the implicit k-fold cross-validation, a steganalysis-aware C-S pair preserving cross-validation should be used. We stress that this is a steganalysis-specific issue and does not indicate any implementation flaw in LIBSVM.

We note that the issue of C-S pairs and its implications for cross-validation procedure in steganalysis has already been pointed out by Schwamberger and Franz in 2010 [121]. We believe, however, that the message may have been hidden to the reader in other experiments and conclusions presented in [121], as the authors studied not only the cross-validation, but also different normalization techniques, and performed numerous experiments using different features and stego-algorithms. This appendix, on the other hand, is devoted solely to this problem. Furthermore, we go more in depth,

provide an explanation, and also study the severity w.r.t. payload. Moreover, we point out a few examples of published works with results affected by the improper cross-validation.

Throughout this appendix, we decided to use SVM as a classifier – it is a popular choice in steganalysis[5] and a known and theoretically well founded classification tool [120]. We did not want to obscure any important insight by involving the novelty of the ensemble classifier, even though the C-S pairing had to be taken into account when developing the ensemble classifier as well (see Chapter 3.1).

C.3.1 Formalization of the k-fold cross-validation in SVM training

Following the notation introduced in Chapter 2.6, the set of features used for classifier training will be denoted $\mathcal{S}^{\text{trn}}$. The k-fold cross-validation divides $\mathcal{S}^{\text{trn}}$ into k disjoint and approximately equally populated groups $\mathcal{S}_i^{\text{trn}}$, $i = 1, \ldots, k$, called folds, where $\mathcal{S}^{\text{trn}} = \cup_i \mathcal{S}_i^{\text{trn}}$ and $\mathcal{S}_i^{\text{trn}} \cap \mathcal{S}_j^{\text{trn}} = \emptyset$ for $i \neq j$. The first fold $\mathcal{S}_1^{\text{trn}}$ is then set apart, the classifier is trained on the union of the remaining $k-1$ folds and its performance is evaluated in terms of the error rate on the samples from $\mathcal{S}_1^{\text{trn}}$ that were not used during the training. This procedure is repeated k times, setting apart subsequently all the folds and using them as an evaluation feedback. All k error-rate estimates are then combined to form the final cross-validation error estimate, E^{cv}, which is an estimate of the real testing error. Typically, k-fold cross-validation is repeated for a pre-defined set of classifier parameters, and the parameters yielding the lowest value of E^{cv} are then chosen for the final classifier training. The parameter search over the grid of values is called *gridsearch*. A more detailed discussion on k-fold cross-validation and gridsearch can be found, for example, in [55].

Through the choice of its *kernel*, the SVM classifier is capable of learning non-linear decision boundaries of desired complexity. More complex decision boundaries allow for accurate fits on the training set, however, they are also more prone to overtraining. Gridsearch with k-fold cross-validation is a common technique for an automated optimization of kernel parameters for the best *testing* performance.

Arguably, the Gaussian SVM (G-SVM) is the most popular kernel choice. G-SVM is parametrized by the misclassification cost C and the kernel width γ,[6] which need to be optimized. Formally, the optimal parameters C^{opt}, γ^{opt} are found as

$$(C^{\text{opt}}, \gamma^{\text{opt}}) = \arg \min_{(C,\gamma) \in \mathcal{G}_C \times \mathcal{G}_\gamma} E^{\text{cv}}(C, \gamma), \tag{C.3.1}$$

where $\mathcal{G}_C \times \mathcal{G}_\gamma$ is a pre-defined grid of parameter values. Once the best parameters $(C^{\text{opt}}, \gamma^{\text{opt}})$ are obtained, they are used to retrain the SVM on the entire training set $\mathcal{S}^{\text{trn}}$. The resulting SVM is ready to be used for real predictions (or for the predictions on the testing set).

C.3.2 Implications for steganalysis

As argued earlier, training sets in steganalysis are characterized by pairs of feature vectors with opposite labels lying close to each other in the feature space $\mathcal{F}$, the so-called cover-stego (C-S) pairs. This pairing seems to be inevitable as steganography advances, even though the goal of steganalysis is to create feature spaces where the cover and stego features are separated as much as possible.

Formally, still following the notation from Chapter 2.6, the steganalysis training set $\mathcal{S}^{\text{trn}}$ consists of two parts, $\mathcal{S}^{\text{trn}} = \mathcal{X}^{\text{trn}} \cup \mathcal{Y}^{\text{trn}}$, where the sets $\mathcal{X}^{\text{trn}} = \{\mathbf{x}_i | i \in \mathcal{I}^{\text{trn}}\}$ and $\mathcal{Y}^{\text{trn}} = \{\mathbf{y}_i | i \in \mathcal{I}^{\text{trn}}\}$ represent the set of cover features, $\mathbf{x}_i \in \mathcal{F}$, and the corresponding stego features, $\mathbf{y}_i \in \mathcal{F}$, respectively, with

[5] The popularity of SVMs could be partially accounted to the LIBSVM [21], a publicly available implementation of this non-trivial learning machinery with a user-friendly interface.

[6] For two features $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{F}$, the Gaussian kernel is defined as $\exp(-\gamma ||\mathbf{x}_i - \mathbf{x}_j||^2)$; smaller γ thus implies a wider kernel.

$\mathcal{I}^{\text{trn}}$, $|\mathcal{I}^{\text{trn}}| = N^{\text{trn}}$, being the number of C-S pairs in the training set. Note that the C-S pairs are preserved.

Importantly, the C-S pairs need to be preserved during the k-fold cross-validation as well! In other words, the C-S pairs cannot be split into two different folds – for the same reason why the C-S pairs need to be preserved when dividing the original image database into training and testing parts. If the C-S pairs were split, the error estimate E^{cv} would be evaluated also on samples whose C-S counterparts were used during the classifier training.

To be more specific, let us assume that the C-S pairing is not taken into account when creating the individual folds for k-fold cross-validation (as in LIBSVM). The probability that the cover's stego counterpart is in the same fold is $\frac{N-k}{k(N-1)}$, where $N = 2N^{\text{trn}}$, which simplifies roughly to $1/k$ if $k \ll N$. For example, for the frequently used five-fold cross-validation, only about 20% of the C-S pairs are preserved. This means that when the performance of SVM is evaluated on the fold $\mathcal{S}_i^{\text{trn}}$, the majority (roughly 80%) of all the examples in $\mathcal{S}_i^{\text{trn}}$ have the second feature from their C-S pair in the set the SVM was trained on! This is certainly undesirable and may negatively influence the results of the cross-validation procedure. In the next section, we will demonstrate the seriousness of this issue.

C.3.3 Experiment

The effect of improperly formed cross-validation sets will be demonstrated on a JPEG domain steganographic algorithm nsF5, briefly described in Appendix A.5. For the purpose of this experiment, we used the BOSSbase image database (see Appendix B), JPEG-compressed with quality factor 75 using Matlab's `imwrite` function. The stego images were created over a range of payloads from 0.01 to 0.20 bpac. For simplicity, we used the 548-dimensional feature space $\mathcal{F}$ formed by CC-PEV features [77].[7]

For every payload, we trained a separate G-SVM on a randomly selected half of the image database, and tested the performance on the other half (C-S pairs are preserved during this division). The SVM was trained using five-fold cross-validation on the following grid of parameters C and γ:

$$(C, \gamma) \in \left\{ \left(10^{\alpha}, \frac{1}{d} 2^{\beta} \right) \middle| \, \alpha = -3, \ldots, 4, \beta = -3, \ldots, 3 \right\}, \tag{C.3.2}$$

where $d = 548$ is the feature space dimensionality. Two different strategies for creating the folds $\mathcal{S}_i^{\text{trn}}$ were used:

1. Implicit cross-validation (as implemented in LIBSVM),
2. Manually created folds preserving C-S pairs.

We note that in [121], the first strategy was called the standard cross-validation, while the second one "paired cross-validation."

Since the purpose of this experiment is to compare the two fold-forming strategies rather than to report a statistically reliable detection performance, our experiment was conducted over a single split of the BOSSbase into training and testing parts. The results for both fold-forming strategies are plotted in Figure C.3.1. We show both P_{E}, the error obtained from the testing set, as well as its estimate E^{cv} coming from the best points of the grid (C.3.2).

There are two patterns to be observed from Figure C.3.1. First, ignoring C-S pairs delivers sub-optimal testing performance over manually created folds that preserve the pairs. This is barely noticeable for large payloads, but becomes more pronounced with decreasing payload, culminating in a "jump" from roughly 43% to undetectability (50%) between 0.03 and 0.04 bpac. The second

[7] Feature extractor is available at `http://dde.binghamton.edu/download/ccmerged/`.

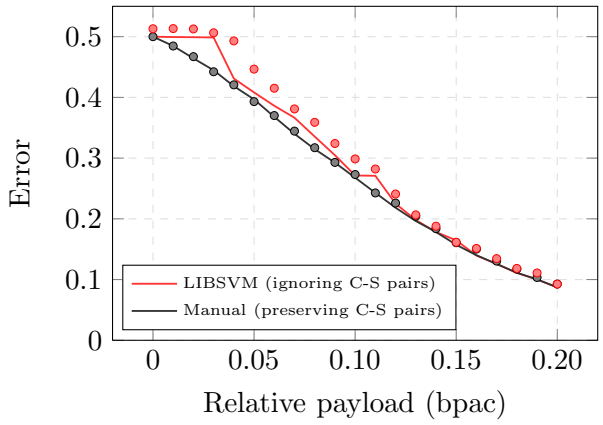


Figure C.3.1: Real testing errors P_{E} (solid line) and the corresponding CV estimates E^{cv} (dots) across different payloads of nsF5 and for two different strategies of forming cross-validation folds.

pattern is a growing overshoot of the error estimate E^{cv} over the real testing error P_{E} as the relative payload decreases. This difference is about 7% at 0.04 bpac and then vanishes when both errors jump to random guessing. Notice, that for very small payloads, the values of E^{cv} are slightly above the random guessing value of 0.5.

C.3.4 Explanation

In the five-fold cross-validation that ignores the C-S pairs, roughly 80% of the validation samples have their C-S counterpart in the set on which the SVM was trained. Figure C.3.2 is a 2D illustration of what happens when the SVM predicts the class labels on such examples. In case of a simple decision boundary (the bottom portion of the figure), the classifier trained on C-S pairs yields a similar predictor as the classifier trained on C-S pairs with missing features. If these missing features are then used for the estimation of the testing error P_{E}, they are classified correctly and the obtained error estimate E^{cv} is indeed a good estimate of the real testing error (for which *all* training C-S pairs are used in the training phase). On the other hand, when the classes are less distinguishable (the case of smaller payloads), the decision boundary learned from all C-S pairs and the one learned from a set consisting of only a single feature from every C-S pair may be quite different, as illustrated in the top part of Figure C.3.2. Most of the missing pairs are then classified incorrectly during validation as they are assigned the label of their counterpart that appeared in the classifier training. Consequently, the CV error E^{cv} is higher (and often *much* higher as will be shown later) than the testing error P_{E}.

The reasoning above explains the growing overshoot of E^{cv} over P_{E} as payload decreases when the incorrect implementation of cross-validation is used – the classes are less distinguishable and the decision boundary is more complex. But this would not be sufficient by itself to explain the different *testing* performance of both types of cross-validation because the *whole* training set is used for the final SVM training in both cases (all C-S pairs are preserved).

If the overshoot was roughly the same for all the points in the grid (C.3.2), the resulting optimal parameters $(C^{\mathrm{opt}}, \gamma^{\mathrm{opt}})$, and thus the testing errors P_{E}, would be the same for both fold-forming strategies, and the only consequence of the incorrect cross-validation would be the inability to accurately estimate the testing error as the overshoot differs from payload to payload. Unfortunately, that is not the case – the complexity of the constructed decision boundary does not depend only on the class distinguishability, but also on the SVM hyper-parameters (C, γ). The larger is the misclassification cost C and the narrower is the Gaussian kernel (the larger is γ), the more complex is the learned class boundary. Therefore, the overshoot is larger for larger values of C and γ.

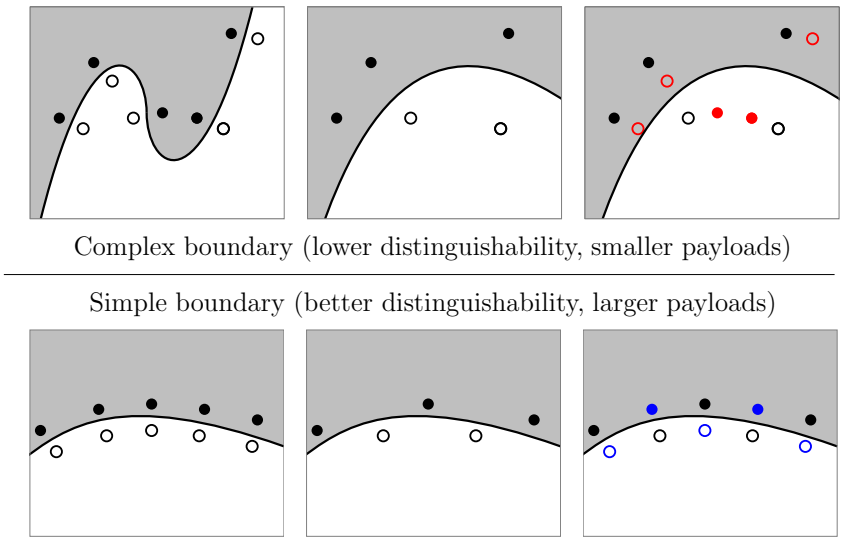


Figure C.3.2: Illustration of what happens when C-S pairs are separated. Top: the case of a complex decision boundary; Bottom: the case of a simple decision boundary. Left: learned boundary when all C-S pairs are included; Middle: learned boundary when one feature from every C-S pair is missing; Right: Correctly (blue) and incorrectly (red) classified points when the missing features are used for testing.

Consequently, during the incorrectly formed cross-validation, the optimal parameters C^{opt} and/or γ^{opt} are being artificially shifted to smaller (and sub-optimal) values as payload decreases, and thus the final testing error is getting higher than it would be if the correct parameters $(C^{\text{opt}}, \gamma^{\text{opt}})$ were found.

We experimentally confirmed these claims and demonstrate them in Figure C.3.3 where we show the results of the grid-search for a fixed small payload of 0.02 bpac. The left part of the figure shows the correctly performed grid-search with the lowest error estimate $E^{\text{cv}} = 0.4671$ found at the point $(C^{\text{opt}}, \gamma^{\text{opt}}) = (10^4, \frac{1}{d}2^{-2})$ which corresponds to the real testing error of nsF5 at this payload for our experimental setup (c.f., Figure C.3.1). On the other hand, when the incorrect cross-validation is performed, the very same point of the grid results in the error estimate $E^{\text{cv}} = 0.7084$, i.e., there is an overshoot of more than 20% (see the point marked with a circle in the right graph of Figure C.3.3). Instead, the "best" point of the grid was declared as the point marked by a cross lying in the random–guessing area in the left part of the grid. Note that the highest overshoot is indeed in the top right part of the grid with higher values of the cost C and with a narrow kernel (large γ) – in the areas where the SVM forms complex decision boundaries. Also note that error rates over 50% are suspicious by themselves as the worst possible error should not exceed 50% (random guessing).

To complete our understanding of the inner workings of the incorrectly performed cross-validation, let us make a closer inspection of the point marked by a circle in Figure C.3.3 ($C = 10^4$ and $\gamma = \frac{1}{d}2^{-2}$). Each of the folds $\mathcal{S}_i^{\text{trn}}, i = 1, \ldots, 5$, consists of two disjoint parts $\mathcal{S}_i^{\text{trn}} = \mathcal{A}_i \cup \mathcal{B}_i$, where the set $\mathcal{A}_i$ is the union of all C-S pairs from $\mathcal{S}_i^{\text{trn}}$ and the set $\mathcal{B}_i$ contains those samples $\mathbf{x} \in \mathcal{S}_i^{\text{trn}}$ whose C-S counterparts appear in a different fold and thus were used for the SVM training. Let $n_{\mathcal{A}_i} = |\mathcal{A}_i|$, $n_{\mathcal{B}_i} = |\mathcal{B}_i|$ be the sizes of sets $\mathcal{A}_i$ and $\mathcal{B}_i$. Let $E^{\text{cv}}_{\mathcal{A}_i}$ and $E^{\text{cv}}_{\mathcal{B}_i}$ be the validation errors obtained only from the sets $\mathcal{A}_i$ and $\mathcal{B}_i$, respectively. Then the cross-validation error obtained from the ith fold is calculated as $E^{\text{cv}}_i = (n_{\mathcal{A}_i} E^{\text{cv}}_{\mathcal{A}_i} + n_{\mathcal{B}_i} E^{\text{cv}}_{\mathcal{B}_i})/(n_{\mathcal{A}_i} + n_{\mathcal{B}_i})$. The values of $n_{\mathcal{A}_i}$, $n_{\mathcal{B}_i}$, $E^{\text{cv}}_{\mathcal{A}_i}$, $E^{\text{cv}}_{\mathcal{B}_i}$ and E^{cv}_i for this particular point of the grid are shown in Table C.3. We can see that while the CV error $E^{\text{cv}}_{\mathcal{A}_i}$ always correctly estimates the real testing error (values around 47%), the error rate $E^{\text{cv}}_{\mathcal{B}_i}$ lies above 75%, confirming that only the validation samples from $\mathcal{B}_i$ (where the C-S pairs are not preserved) are responsible for the error overshoot. Since roughly 80% of the validation samples are

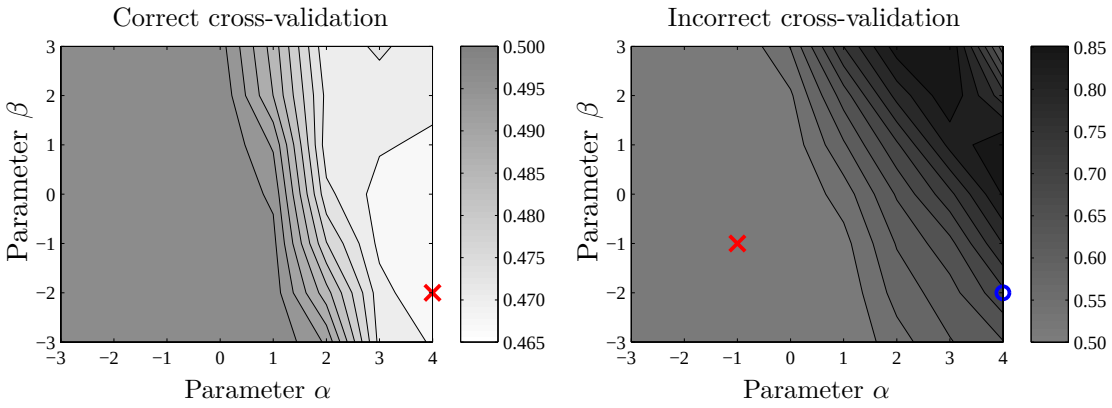


Figure C.3.3: Contour graphs of the grid-search error estimates E^{cv} as functions of the parameters $C = 10^{\alpha}$ and $\gamma = \frac{1}{d}2^{\beta}$ at payload 0.02 bpac for both correctly (left) and incorrectly (right) performed cross-validation. The crosses mark the points with the lowest CV errors. The circle marks the point commented on in the text.

Fold i	Size $\lvert\mathcal{S}_i^{\text{trn}}\rvert$	$n_{\mathcal{A}_i}$	$n_{\mathcal{B}_i}$	$E^{\text{cv}}_{\mathcal{A}_i}$	$E^{\text{cv}}_{\mathcal{B}_i}$	E^{cv}_i
$\mathcal{S}_1^{\text{trn}}$	2001	394	1607	.4772	.7561	.7011
$\mathcal{S}_2^{\text{trn}}$	1980	398	1582	.4799	.7794	.7192
$\mathcal{S}_3^{\text{trn}}$	2047	406	1641	.4754	.7782	.7181
$\mathcal{S}_4^{\text{trn}}$	1990	430	1560	.4744	.7750	.7101
$\mathcal{S}_5^{\text{trn}}$	1982	388	1594	.4665	.7501	.6937
Mean	2000	403.2	1596.8	.4747	.7676	.7084

Table C.3: Results of the incorrect five-fold cross-validation at the grid point $(10^4, \frac{1}{d}2^{-2})$ – the point marked with a circle in Figure C.3.3. All symbols are defined in the text.

from $\mathcal{B}_i$, the resulting error $E_i^{\text{cv}} \approx 70\%$. This is in agreement with the 2D illustration in Figure C.3.2 (top) – the red points correspond to the artificially misclassified samples from $\mathcal{B}_i$.

C.3.5 Summary

We showed that the standard k-fold cross-validation procedure, as implemented for example in the popular machine learning package LIBSVM, should not be used for steganalysis as it does not preserve the pairs of cover-stego features. Instead, manually created folds taking this specifics of steganalysis into account should be used, as incorrectly created folds result in misleading values of error estimates and consequently in a suboptimal performance. The negative impact of incorrectly executed cross-validation manifests itself stronger when the class distinguishability is lower, i.e., for smaller payloads. This makes the whole problem even more important as the secure payload of a given steganographic algorithm is defined as the maximal payload that can be embedded without being detected (the best possible detector is a random guesser). For example, ignoring cover-stego pairs, one might conclude that the secure payload of nsF5 is around 0.03 bpac (see Figure C.3.1). However, that would be an incorrect conclusion as we can detect nsF5 at this payload with error around 43% if we take the cover-stego pairing into account.

We would like to conclude this appendix by pointing out a few examples of published works with results that may have been affected by the improper cross-validation. All of the following publications

exhibit a "jump" in the reported performance to random guessing, similar to the one in Figure C.3.1: [35] (Figures 3 and 4), [36] (Figures 10 and 12), [32] (Figure 6), [34] (Figure 4), [33] (Figure 2). Note that the list contains only those publications where the results are reported graphically and where the focus is on smaller payloads as there the problem is "visible."

Bibliography

[1] H. Ahn, H. Moon, M. J. Fazzari, N. Lim, J. J. Chen, and R. L. Kodell. Classification by ensembles from random partitions of high-dimensional data. *Comput. Stat. Data Anal.*, 51:6166–6179, August 2007.

[2] N. Asadi, M. Jamzad, and H. Sajedi. Improvements of image-steganalysis using boosted combinatorial classifiers and gaussian high pass filtering. In *Proceedings of the 2008 International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, IIH-MSP '08, pages 1508–1511, Washington, DC, USA, 2008. IEEE Computer Society.

[3] A. Assareh, M. H. Moradi, and L. G. Volkert. A hybrid random subspace classifier fusion approach for protein mass spectra classification. In *Proceedings of the 6th European conference on Evolutionary computation, machine learning and data mining in bioinformatics*, EvoBIO'08, pages 1–11. Springer-Verlag, Berlin, Heidelberg, 2008.

[4] F. Bacon. *Of the Advancement and Proficiencie of Learning or the Partitions of Sciences*, volume VI. Leon Lichfield, Oxford, for R. Young and E. Forest, 1640.

[5] İ. Avcıbaş, M. Kharrazi, N. D. Memon, and B. Sankur. Image steganalysis with binary similarity measures. *EURASIP Journal on Applied Signal Processing*, 17:2749–2757, 2005.

[6] İ. Avcıbaş, N. D. Memon, and B. Sankur. Steganalysis using image quality metrics. In E. J. Delp and P. W. Wong, editors, *Proceedings SPIE, Electronic Imaging, Security and Watermarking of Multimedia Contents III*, volume 4314, pages 523–531, San Jose, CA, January 22–25, 2001.

[7] İ. Avcıbaş, N. D. Memon, and B. Sankur. Image steganalysis with binary similarity measures. In *Proceedings IEEE, International Conference on Image Processing, ICIP 2002*, volume 3, pages 645–648, Rochester, NY, September 22–25, 2002.

[8] E. Barkan, E. Biham, and N. Keller. Instant ciphertext-only cryptanalysis of gsm encrypted communication. *Journal of Cryptology*, 21:392–429, March 2008.

[9] P. Bas, T. Filler, and T. Pevný. Break our steganographic system – the ins and outs of organizing BOSS. In T. Filler, T. Pevný, A. Ker, and S. Craver, editors, *Information Hiding, 13th International Workshop*, volume 6958 of Lecture Notes in Computer Science, pages 59–70, Prague, Czech Republic, May 18–20, 2011.

[10] P. Bas and T. Furon. Break Our Watermarking System 2nd Ed., BOWS-2. http://bows2.ec-lille.fr, July 2007.

[11] R. Böhme. *Improved Statistical Steganalysis Using Models of Heterogeneous Cover Signals*. PhD thesis, Faculty of Computer Science, Technische Universität Dresden, Germany, 2008.

[12] J. Brassil, S. Low, N. F. Maxemchuk, and L. O'Gorman. Hiding information in document images. In *Proceedings of the Conference on Information Sciences and Systems, CISS*, pages 482–489, Johns Hopkins University, Baltimore, MD, March 22–24, 1995.

[13] L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, August 1996.

[14] L. Breiman. Random forests. *Machine Learning*, 45:5–32, October 2001.

[15] D. Brewster. *Microscope*, volume XIV. Encyclopaedia Britannica or the Dictionary of Arts, Sciences, and General Literature, Edinburgh, IX – Application of photography to the microscope, 8th edition, pages 801–802, 1857.

[16] G. Brown. An information theoretic perspective on multiple classifier systems. In J. Benediktsson, J. Kittler, and F. Roli, editors, *Multiple Classifier Systems*, volume 5519 of *Lecture Notes in Computer Science*, pages 344–353. Springer Berlin, Heidelberg, 2009.

[17] R. Bryll, R. Gutierrez-Osuna, and F. Quek. Attribute bagging: Improving accuracy of classifier ensembles by using random feature subsets. *Pattern Recognition*, 36(6):1291–1302+, 2003.

[18] I. Burrell. Fifty held in worldwide paedophile crackdown. *The Independent*, July 3, 2002.

[19] C. Cachin. An information-theoretic model for steganography. In D. Aucsmith, editor, *Information Hiding, 2nd International Workshop*, volume 1525 of Lecture Notes in Computer Science, pages 306–318, Portland, OR, April 14–17, 1998. Springer-Verlag, New York.

[20] J. Carr. Anti-forensic methods used by Jihadist web sites. *eSecurity Planet*, August 16, 2007.

[21] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[22] C. Chen and Y. Q. Shi. JPEG image steganalysis utilizing both intrablock and interblock correlations. In *Circuits and Systems, ISCAS 2008. IEEE International Symposium on*, pages 3029–3032, May 2008.

[23] V. Chonev and A. D. Ker. Feature restoration and distortion metrics. In A. Alattar, N. D. Memon, E. J. Delp, and J. Dittmann, editors, *Proceedings SPIE, Electronic Imaging, Media Watermarking, Security and Forensics of Multimedia XIII*, volume 7880, pages 0G01–0G14, San Francisco, CA, January 23–26, 2011.

[24] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. New York: John Wiley & Sons, Inc., 1991.

[25] A. Dasgupta. Mumbai police fail to crack July 11 suspects' mail. *Daily News & Analysis*, October 16, 2006.

[26] J. Davidson and J. Jalan. Steganalysis using partially ordered Markov models. In R. Böhme and R. Safavi-Naini, editors, *Information Hiding, 12th International Workshop*, volume 6387 of Lecture Notes in Computer Science, pages 118–132, Calgary, Canada, June 28–30, 2010. Springer-Verlag, New York.

[27] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. New York: John Wiley & Sons, Inc., 2nd edition, 2001.

[28] S. Dumitrescu, X. Wu, and N. D. Memon. On steganalysis of random LSB embedding in continuous-tone images. In *Proceedings IEEE, International Conference on Image Processing, ICIP 2002*, pages 324–339, Rochester, NY, September 22–25, 2002.

[29] R. Engel. Al-Qaida spreads across the web. *NBC Nightly News*, August 21, 2007.

[30] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008. Software available at http://www.csie.ntu.edu.tw/~cjlin/liblinear.

[31] H. Farid and L. Siwei. Detecting hidden messages using higher-order statistics and support vector machines. In F. A. P. Petitcolas, editor, *Information Hiding, 5th International Workshop*, volume 2578 of Lecture Notes in Computer Science, pages 340–354, Noordwijkerhout, The Netherlands, October 7–9, 2002. Springer-Verlag, New York.

[32] T. Filler and J. Fridrich. Gibbs construction in steganography. *IEEE Transactions on Information Forensics and Security*, 5(4):705–720, 2010.

[33] T. Filler and J. Fridrich. Minimizing additive distortion functions with non-binary embedding operation in steganography. In *Information Forensics and Security (WIFS), 2010 IEEE International Workshop on*, pages 1–6, December 2010.

[34] T. Filler and J. Fridrich. Steganography using Gibbs random fields. In J. Dittmann and S. Craver, editors, *Proceedings of the 12th ACM Multimedia & Security Workshop*, pages 199–212, Rome, Italy, September 9–10, 2010. ACM.

[35] T. Filler and J. Fridrich. Design of adaptive steganographic schemes for digital images. In A. Alattar, N. D. Memon, E. J. Delp, and J. Dittmann, editors, *Proceedings SPIE, Electronic Imaging, Media Watermarking, Security and Forensics of Multimedia XIII*, volume 7880, pages OF 1–14, San Francisco, CA, January 23–26, 2011.

[36] T. Filler, J. Judas, and J. Fridrich. Minimizing additive distortion in steganography using syndrome-trellis codes. *IEEE Transactions on Information Forensics and Security*, 6:920–935, 2011.

[37] T. Filler, A. D. Ker, and J. Fridrich. The Square Root Law of steganographic capacity for Markov covers. In N. D. Memon, E. J. Delp, P. W. Wong, and J. Dittmann, editors, *Proceedings SPIE, Electronic Imaging, Security and Forensics of Multimedia XI*, volume 7254, pages 08 1–08 11, San Jose, CA, January 18–21, 2009.

[38] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, pages 23–37, London, UK, 1995. Springer-Verlag.

[39] J. Fridrich. Feature-based steganalysis for JPEG images and its implications for future design of steganographic schemes. In J. Fridrich, editor, *Information Hiding, 6th International Workshop*, volume 3200 of Lecture Notes in Computer Science, pages 67–81, Toronto, Canada, May 23–25, 2004. Springer-Verlag, New York.

[40] J. Fridrich. *Steganography in Digital Media: Principles, Algorithms, and Applications*. Cambridge University Press, 2009.

[41] J. Fridrich and M. Goljan. On estimation of secret message length in LSB steganography in spatial domain. In E. J. Delp and P. W. Wong, editors, *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VI*, volume 5306, pages 23–34, San Jose, CA, January 19–22, 2004.

[42] J. Fridrich, M. Goljan, and R. Du. Reliable detection of LSB steganography in grayscale and color images. In J. Dittmann, K. Nahrstedt, and P. Wohlmacher, editors, *Proceedings of the ACM, Special Session on Multimedia Security and Watermarking*, pages 27–30, Ottawa, Canada, October 5, 2001.

[43] J. Fridrich, M. Goljan, and D. Hogea. Attacking the OutGuess. In *Proceedings of the ACM, Special Session on Multimedia Security and Watermarking*, Juan-les-Pins, France, December 6, 2002.

[44] J. Fridrich, M. Goljan, and D. Hogea. Steganalysis of JPEG images: Breaking the F5 algorithm. In *Information Hiding, 5th International Workshop*, volume 2578 of Lecture Notes in Computer Science, pages 310–323, Noordwijkerhout, The Netherlands, October 7–9, 2002. Springer-Verlag, New York.

[45] J. Fridrich, M. Goljan, D. Hogea, and D. Soukal. Quantitative steganalysis of digital images: Estimating the secret message length. *ACM Multimedia Systems Journal*, 9(3):288–302, 2003.

[46] J. Fridrich, M. Goljan, and D. Soukal. Perturbed quantization steganography using wet paper codes. In J. Dittmann and J. Fridrich, editors, *Proceedings of the 6th ACM Multimedia & Security Workshop*, pages 4–15, Magdeburg, Germany, September 20–21, 2004.

[47] J. Fridrich, M. Goljan, and D. Soukal. Wet paper codes with improved embedding efficiency. *IEEE Transactions on Information Forensics and Security*, 1(1):102–110, 2006.

[48] J. Fridrich and J. Kodovský. Rich models for steganalysis of digital images. *IEEE Transactions on Information Forensics and Security*. To appear.

[49] J. Fridrich, J. Kodovský, M. Goljan, and V. Holub. Breaking HUGO – the process discovery. In T. Filler, T. Pevný, A. Ker, and S. Craver, editors, *Information Hiding, 13th International Workshop*, volume 6958 of Lecture Notes in Computer Science, pages 85–101, Prague, Czech Republic, May 18–20, 2011.

[50] J. Fridrich, J. Kodovský, M. Goljan, and V. Holub. Steganalysis of content-adaptive steganography in spatial domain. In T. Filler, T. Pevný, A. Ker, and S. Craver, editors, *Information Hiding, 13th International Workshop*, volume 6958 of Lecture Notes in Computer Science, pages 102–117, Prague, Czech Republic, May 18–20, 2011.

[51] J. Fridrich, T. Pevný, and J. Kodovský. Statistically undetectable JPEG steganography: Dead ends, challenges, and opportunities. In J. Dittmann and J. Fridrich, editors, *Proceedings of the 9th ACM Multimedia & Security Workshop*, pages 3–14, Dallas, TX, September 20–21, 2007.

[52] M. Goljan, J. Fridrich, and T. Holotyak. New blind steganalysis and its implications. In E. J. Delp and P. W. Wong, editors, *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VIII*, volume 6072, pages 1–13, San Jose, CA, January 16–19, 2006.

[53] Q. Guan, J. Dong, and T. Tan. Blind quantitative steganalysis based on feature fusion and gradient boosting. In H.-J. Kim, Y. Shi, and M. Barni, editors, *Digital Watermarking*, volume 6526 of *Lecture Notes in Computer Science*, pages 266–279. Springer Berlin, Heidelberg, 2011.

[54] G. Gül and F. Kurugollu. A new methodology in steganalysis: Breaking highly undetactable steganograpy (HUGO). In T. Filler, T. Pevný, A. Ker, and S. Craver, editors, *Information Hiding, 13th International Workshop*, volume 6958 of Lecture Notes in Computer Science, pages 71–84, Prague, Czech Republic, May 18–20, 2011.

[55] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition (Springer Series in Statistics)*. Springer-Verlag, 2009.

[56] Herodotus. *The Histories*. Penguin Books, London, 1996. Translated by Aubrey de Sélincourt.

[57] S. Hetzl and P. Mutzel. A graph-theoretic approach to steganography. In J. Dittmann, S. Katzenbeisser, and A. Uhl, editors, *Communications and Multimedia Security, 9th IFIP TC-6 TC-11 International Conference, CMS 2005*, volume 3677 of Lecture Notes in Computer Science, pages 119–128, Salzburg, Austria, September 19–21, 2005.

[58] T. K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:832–844, 1998.

[59] F. Huang, J. Huang, and B. Li. Universal JPEG steganalysis based on microscopic and macroscopic calibration. In *Proceedings IEEE, International Conference on Image Processing, ICIP 2008*, volume 52, pages 2068–2071, San Diego, CA, October 12–15, 2008.

[60] N. F. Johnson and P. Sallee. Detection of hidden information, covert channels and information flows. In John G. Voeller, editor, *Wiley Handbook of Science Technology for Homeland Security*. New York: Wiley & Sons, Inc, April 4, 2008.

[61] J. Kelley. Terrorist instructions hidden online. *USA Today*, February 5, 2001.

[62] A. D. Ker. Improved detection of LSB steganography in grayscale images. In J. Fridrich, editor, *Information Hiding, 6th International Workshop*, volume 3200 of Lecture Notes in Computer Science, pages 97–115, Toronto, Canada, May 23–25, 2004. Springer-Verlag, Berlin.

[63] A. D. Ker. Resampling and the detection of LSB matching in color bitmaps. In E. J. Delp and P. W. Wong, editors, *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VII*, volume 5681, pages 1–15, San Jose, CA, January 16–20, 2005.

[64] A. D. Ker. Steganalysis of LSB matching in grayscale images. *IEEE Signal Processing Letters*, 12(6):441–444, June 2005.

[65] A. D. Ker. Fourth-order structural steganalysis and analysis of cover assumptions. In E. J. Delp and P. W. Wong, editors, *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VIII*, volume 6072, pages 25–38, San Jose, CA, January 16–19, 2006.

[66] A. D. Ker. Derivation of error distribution in least squares steganalysis. *IEEE Transactions on Information Forensics and Security*, 2:140–148, 2007.

[67] A. D. Ker. Optimally weighted least-squares steganalysis. In E. J. Delp and P. W. Wong, editors, *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents IX*, volume 6505, pages 6 1–6 16, San Jose, CA, January 29–February 1, 2007.

[68] A. D. Ker and R. Böhme. Revisiting weighted stego-image steganalysis. In E. J. Delp, P. W. Wong, J. Dittmann, and N. D. Memon, editors, *Proceedings SPIE, Electronic Imaging, Security, Forensics, Steganography, and Watermarking of Multimedia Contents X*, volume 6819, pages 5 1–5 17, San Jose, CA, January 27–31, 2008.

[69] A. D. Ker, T. Pevný, J. Kodovský, and J. Fridrich. The Square Root Law of steganographic capacity. In A. D. Ker, J. Dittmann, and J. Fridrich, editors, *Proceedings of the 10th ACM Multimedia & Security Workshop*, pages 107–116, Oxford, UK, September 22–23, 2008.

[70] M. Kharrazi, H. T. Sencar, and N. Memon. Improving steganalysis by fusion techniques: A case study with image steganography. *LNCS Transactions on Data Hiding and Multimedia Security I*, 4300:123–137, 2006.

[71] M. Kharrazi, H. T. Sencar, and N. D. Memon. Benchmarking steganographic and steganalytic techniques. In E. J. Delp and P. W. Wong, editors, *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VII*, volume 5681, pages 252–263, San Jose, CA, January 16–20, 2005.

[72] Y. Kim, Z. Duric, and D. Richards. Modified matrix encoding technique for minimal distortion steganography. In J. L. Camenisch, C. S. Collberg, N. F. Johnson, and P. Sallee, editors, *Information Hiding, 8th International Workshop*, volume 4437 of Lecture Notes in Computer Science, pages 314–327, Alexandria, VA, July 10–12, 2006. Springer-Verlag, New York.

[73] M. Kirchner and J. Fridrich. On detection of median filtering in images. In *Proc. SPIE, Electronic Imaging, Media Forensics and Security XII*, volume 7542, pages 10 1–12, San Jose, CA, January 17–21 2010.

[74] J. Kodovský. On dangers of cross-validation in steganalysis. Technical report, Binghamton University, August 2011.

[75] J. Kodovský and J. Fridrich. Influence of embedding strategies on security of steganographic methods in the JPEG domain. In E. J. Delp, P. W. Wong, J. Dittmann, and N. D. Memon, editors, *Proceedings SPIE, Electronic Imaging, Security, Forensics, Steganography, and Watermarking of Multimedia Contents X*, volume 6819, pages 2 1–2 13, San Jose, CA, January 27–31, 2008.

[76] J. Kodovský and J. Fridrich. On completeness of feature spaces in blind steganalysis. In A. D. Ker, J. Dittmann, and J. Fridrich, editors, *Proceedings of the 10th ACM Multimedia & Security Workshop*, pages 123–132, Oxford, UK, September 22–23, 2008.

[77] J. Kodovský and J. Fridrich. Calibration revisited. In J. Dittmann, S. Craver, and J. Fridrich, editors, *Proceedings of the 11th ACM Multimedia & Security Workshop*, pages 63–74, Princeton, NJ, September 7–8, 2009.

[78] J. Kodovský and J. Fridrich. Steganalysis in high dimensions: Fusing classifiers built on random subspaces. In A. Alattar, N. D. Memon, E. J. Delp, and J. Dittmann, editors, *Proceedings SPIE, Electronic Imaging, Media Watermarking, Security and Forensics of Multimedia XIII*, volume 7880, pages OL 1–13, San Francisco, CA, January 23–26, 2011.

[79] J. Kodovský and J. Fridrich. Steganalysis of JPEG images using rich models. In A. Alattar, N. D. Memon, and E. J. Delp, editors, *Proceedings SPIE, Electronic Imaging, Media Watermarking, Security, and Forensics of Multimedia XIV*, San Francisco, CA, January 22–26, 2012.

[80] J. Kodovský, J. Fridrich, and V. Holub. On dangers of overtraining steganography to incomplete cover model. In S. Craver and J. Dittmann, editors, *Proceedings of the 13th ACM Multimedia & Security Workshop*, pages 69–76, Niagara Falls, NY, September 29–30, 2011.

[81] J. Kodovský, J. Fridrich, and V. Holub. Ensemble classifiers for steganalysis of digital media. *IEEE Transactions on Information Forensics and Security*, 7(2):432–444, April 2012.

[82] J. Kodovský, T. Pevný, and J. Fridrich. Modern steganalysis can detect YASS. In N. D. Memon, E. J. Delp, P. W. Wong, and J. Dittmann, editors, *Proceedings SPIE, Electronic Imaging, Security and Forensics of Multimedia XII*, volume 7541, pages 02–01–02–11, San Jose, CA, January 17–21, 2010.

[83] G. Kolata. Veiled messages of terror may lurk in cyberspace. *The New York Times*, October 30, 2001.

[84] T. G. Kolda, R. M. Lewis, and V. Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, 45:385–482, 2003.

[85] L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.

[86] L. I. Kuncheva, J. J. Rodriguez Diez, C. O. Plumpton, D. E. J. Linden, and S. J. Johnston. Random subspace ensembles for fmri classification. *IEEE Transactions on Medical Imaging*, 29:531–542, 2010.

[87] T. N. Lal, O. Chapelle, J. Weston, and A. Elisseeff. Embedded methods. In I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, editors, *Feature Extraction: Foundations and Applications, Studies in Fuzziness and Soft Computing*, pages 137–165. Physica-Verlag, Springer, 2006.

[88] I. Laptev. Improving object detection with boosted histograms. *Journal of Image and Vision Computing*, 27:535–544, April 2009.

[89] K. Lee and A. Westfeld. Generalized category attack – improving histogram-based attack on JPEG LSB embedding. In T. Furon, F. Cayre, G. Doërr, and P. Bas, editors, *Information Hiding, 9th International Workshop*, volume 4567 of Lecture Notes in Computer Science, pages 378–392, Saint Malo, France, June 11–13, 2007. Springer-Verlag, Berlin.

[90] B. Li, Y. Q. Shi, and J. Huang. Steganalysis of YASS. In A. D. Ker, J. Dittmann, and J. Fridrich, editors, *Proceedings of the 10th ACM Multimedia & Security Workshop*, pages 139–148, Oxford, UK, September 22–23, 2008.

[91] Q. Liu. Detection of misaligned cropping and recompression with the same quantization matrix and relevant forgery. In *Proceedings of the 3rd international ACM workshop on Multimedia in forensics and intelligence*, MiFor '11, pages 25–30, New York, NY, USA, 2011. ACM.

[92] Q. Liu. Steganalysis of DCT-embedding based adaptive steganography and YASS. In S. Craver and J. Dittmann, editors, *Proceedings of the 13th ACM Multimedia & Security Workshop*, pages 77–86, Niagara Falls, NY, September 29–30, 2011.

[93] Q. Liu, A. H. Sung, M. Qiao, Z. Chen, and B. Ribeiro. An improved approach to steganalysis of JPEG images. *Information Sciences*, 180(9):1643–1655, 2010.

[94] W. Luo, F. Huang, and J. Huang. Edge adaptive image steganography based on LSB matching revisited. *IEEE Transactions on Information Forensics and Security*, 5(2):201–214, June 2010.

[95] S. Lyu and H. Farid. Steganalysis using color wavelet statistics and one-class support vector machines. In E. J. Delp and P. W. Wong, editors, *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VI*, volume 5306, pages 35–45, San Jose, CA, January 19–22, 2004.

[96] S. Lyu and H. Farid. Steganalysis using higher-order image statistics. *IEEE Transactions on Information Forensics and Security*, 1(1):111–119, 2006.

[97] W. Mazurczyk and K. Szczypiorski. Steganography of VoIP streams. In Robert Meersman and Zahir Tari, editors, *On the Move to Meaningful Internet Systems: OTM 2008*, volume 5332 of *Lecture Notes in Computer Science*, pages 1001–1018. Springer Berlin, Heidelberg, 2008.

[98] C. Mellor. How long does it take take to crack a terrorist hard drive? *TechWorld*, November 4, 2005.

[99] J. Meynet and J.-P. Thiran. Information theoretic combination of classifiers with application to AdaBoost. In M. Haindl, J. Kittler, and F. Roli, editors, *Multiple Classifier Systems*, volume 4472 of *Lecture Notes in Computer Science*, pages 171–179. Springer Berlin, Heidelberg, 2007.

[100] S. Mika, G. Ratsch, J. Weston, B. Schölkopf, and K.R. Mullers. Fisher discriminant analysis with kernels. In *Proceedings of the 1999 IEEE Signal Processing Society Workshop Neural Networks for Signal Processing IX*, pages 41–48, Madison, WI, August 1999.

[101] D. Montgomery. Arrests of alleged spies draws attention to long obscure field of steganography. *The Washington Post*, June 30, 2010.

[102] S. Murdoch and S. Lewis. Embedding covert channels into TCP/IP. In *Information Hiding*, volume 3727 of *Lecture Notes in Computer Science*, pages 247–261. Springer Berlin, Heidelberg, 2005.

[103] F. Pereira and G. Gordon. The support vector decomposition machine. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, pages 689–696, Pittsburgh, PA, 2006.

[104] T. Pevný, P. Bas, and J. Fridrich. Steganalysis by subtractive pixel adjacency matrix. *IEEE Transactions on Information Forensics and Security*, 5(2):215–224, June 2010.

[105] T. Pevný, T. Filler, and P. Bas. Using high-dimensional image models to perform highly undetectable steganography. In R. Böhme and R. Safavi-Naini, editors, *Information Hiding, 12th International Workshop*, volume 6387 of Lecture Notes in Computer Science, pages 161–177, Calgary, Canada, June 28–30, 2010. Springer-Verlag, New York.

[106] T. Pevný and J. Fridrich. Towards multi-class blind steganalyzer for JPEG images. In M. Barni, I. J. Cox, T. Kalker, and H. J. Kim, editors, *International Workshop on Digital Watermarking*, volume 3710 of Lecture Notes in Computer Science, Siena, Italy, September 15–17, 2005. Springer-Verlag, Berlin.

[107] T. Pevný and J. Fridrich. Merging Markov and DCT features for multi-class JPEG steganalysis. In E. J. Delp and P. W. Wong, editors, *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents IX*, volume 6505, pages 3 1–3 14, San Jose, CA, January 29–February 1, 2007.

[108] T. Pevný and J. Fridrich. Benchmarking for steganography. In K. Solanki, K. Sullivan, and U. Madhow, editors, *Information Hiding, 10th International Workshop*, volume 5284 of Lecture Notes in Computer Science, pages 251–267, Santa Barbara, CA, June 19–21, 2008. Springer-Verlag, New York.

[109] T. Pevný and J. Fridrich. Estimation of primary quantization matrix for steganalysis of double-compressed JPEG images. In E. J. Delp, P. W. Wong, J. Dittmann, and N. D. Memon, editors, *Proceedings SPIE, Electronic Imaging, Security, Forensics, Steganography, and Watermarking of Multimedia Contents X*, volume 6819, pages 11 1–11 13, San Jose, CA, January 27–31, 2008.

[110] T. Pevný and J. Fridrich. Multiclass detector of current steganographic methods for JPEG format. *IEEE Transactions on Information Forensics and Security*, 3(4):635–650, December 2008.

[111] T. Pevný and J. Fridrich. Novelty detection in blind steganalysis. In A. D. Ker, J. Dittmann, and J. Fridrich, editors, *Proceedings of the 10th ACM Multimedia & Security Workshop*, pages 167–176, Oxford, UK, September 22–23, 2008.

[112] T. Pevný, J. Fridrich, and A. D. Ker. From blind to quantitative steganalysis. In N. D. Memon, E. J. Delp, P. W. Wong, and J. Dittmann, editors, *Proceedings SPIE, Electronic Imaging, Security and Forensics of Multimedia XI*, volume 7254, pages 0C 1–0C 14, San Jose, CA, January 18–21, 2009.

[113] N. Provos. Defending against statistical steganalysis. In *10th USENIX Security Symposium*, pages 323–335, Washington, DC, August 13–17, 2001.

[114] D. Radcliff. Quickstudy: Steganography: Hidden data. *ComputerWorld*, June 10, 2002.

[115] V. Sachnev, H. J. Kim, and R. Zhang. Less detectable JPEG steganography method based on heuristic optimization and BCH syndrome coding. In J. Dittmann, S. Craver, and J. Fridrich, editors, *Proceedings of the 11th ACM Multimedia & Security Workshop*, pages 131–140, Princeton, NJ, September 7–8, 2009.

[116] P. Sallee. Model-based steganography. In T. Kalker, I. J. Cox, and Y. Man Ro, editors, *Digital Watermarking, 2nd International Workshop*, volume 2939 of Lecture Notes in Computer Science, pages 154–167, Seoul, Korea, October 20–22, 2003. Springer-Verlag, New York.

[117] P. Sallee. Model-based methods for steganography and steganalysis. *International Journal of Image Graphics*, 5(1):167–190, 2005.

[118] A. Sarkar, K. Solanki, and B. S. Manjunath. Further study on YASS: Steganography based on randomized embedding to resist blind steganalysis. In E. J. Delp, P. W. Wong, J. Dittmann, and N. D. Memon, editors, *Proceedings SPIE, Electronic Imaging, Security, Forensics, Steganography, and Watermarking of Multimedia Contents X*, volume 6819, pages 16–31, San Jose, CA, January 27–31, 2008.

[119] R. E. Schapire. The boosting approach to machine learning: An overview. In D. D. Denison, M. H. Hansen, C. Holmes, B. Mallick, and B. Yu, editors, *Nonlinear Estimation and Classification*. Springer, 2003.

[120] B. Schölkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning)*. The MIT Press, 2001.

[121] V. Schwamberger and M. O. Franz. Simple algorithmic modifications for improving blind steganalysis performance. In J. Dittmann and S. Craver, editors, *Proceedings of the 12th ACM Multimedia & Security Workshop*, pages 225–230, Rome, Italy, September 9–10, 2010.

[122] G. Serpen and S. Pathical. Classification in high-dimensional feature spaces: Random subsample ensemble. In *Machine Learning and Applications, 2009. ICMLA '09. International Conference on*, pages 740 –745, December 2009.

[123] Y. Q. Shi, C. Chen, and W. Chen. A Markov process based approach to effective attacking JPEG steganography. In J. L. Camenisch, C. S. Collberg, N. F. Johnson, and P. Sallee, editors, *Information Hiding, 8th International Workshop*, volume 4437 of Lecture Notes in Computer Science, pages 249–264, Alexandria, VA, July 10–12, 2006. Springer-Verlag, New York.

[124] G. J. Simmons. The prisoner's problem and the subliminal channel. In D. Chaum, editor, *Advances in Cryptology, CRYPTO '83*, pages 51–67, Santa Barbara, CA, August 22–24, 1983. New York: Plenum Press.

[125] K. Solanki, N. Jacobsen, U. Madhow, B. S. Manjunath, and S. Chandrasekaran. Robust image-adaptive data hiding based on erasure and error correction. *IEEE Transactions on Image Processing*, 13(12):1627–1639, December 2004.

[126] K. Solanki, A. Sarkar, and B. S. Manjunath. YASS: Yet another steganographic scheme that resists blind steganalysis. In T. Furon, F. Cayre, G. Doërr, and P. Bas, editors, *Information Hiding, 9th International Workshop*, volume 4567 of Lecture Notes in Computer Science, pages 16–31, Saint Malo, France, June 11–13, 2007. Springer-Verlag, New York.

[127] K. Solanki, K. Sullivan, U. Madhow, B. S. Manjunath, and S. Chandrasekaran. Provably secure steganography: Achieving zero K–L divergence using statistical restoration. In *Proceedings IEEE, International Conference on Image Processing, ICIP 2006*, pages 125–128, Atlanta, GA, October 8–11, 2006.

[128] A. Tacticius. *How to Survive Under Siege/Aineas the Tactician*. Oxford: Clarendon Ancient History Series, 1990.

[129] D. Upham. Steganographic algorithm JSteg. Software available at `http://zooid.org/~paul/crypto/jsteg`.

[130] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

[131] Y. Wang, JF. Liu, and WM. Zhang. Blind JPEG steganalysis based on correlations of DCT cofficients in multi-directions and calibrations. In *Proceedings of the 2009 International Conference on Multimedia Information Networking and Security - Volume 01*, MINES '09, pages 495–499, Washington, DC, USA, 2009. IEEE Computer Society.

[132] A. Westfeld. High capacity despite better steganalysis (F5 – a steganographic algorithm). In I. S. Moskowitz, editor, *Information Hiding, 4th International Workshop*, volume 2137 of Lecture Notes in Computer Science, pages 289–302, Pittsburgh, PA, April 25–27, 2001. Springer-Verlag, New York.

[133] A. Westfeld and A. Pfitzmann. Attacks on steganographic systems. In A. Pfitzmann, editor, *Information Hiding, 3rd International Workshop*, volume 1768 of Lecture Notes in Computer Science, pages 61–75, Dresden, Germany, September 29–October 1, 1999. Springer-Verlag, New York.

[134] E. H. Wilkins. *A History of Italian Literature*. Oxford University Press, London, 1954.

[135] G. Xuan, Y. Q. Shi, J. Gao, D. Zou, C. Yang, Z. Z. P. Chai, C. Chen, and W. Chen. Steganalysis based on multiple features formed by statistical moments of wavelet characteristic functions. In M. Barni, J. Herrera, S. Katzenbeisser, and F. Pérez-González, editors, *Information Hiding, 7th International Workshop*, volume 3727 of Lecture Notes in Computer Science, pages 262–277, Barcelona, Spain, June 6–8, 2005. Springer-Verlag, Berlin.

[136] Z. Yu and H.-S. Wong. Image classification based on the bagging-AdaBoost ensemble. In *Proceedings IEEE, International Conference on Multimedia and Expo*, pages 1481–1484, Hannover, Germany, June 23–April 26, 2008.

[137] D. Zou, Y. Q. Shi, W. Su, and G. Xuan. Steganalysis based on Markov model of thresholded prediction-error image. In *Proceedings IEEE, International Conference on Multimedia and Expo*, pages 1365–1368, Toronto, Canada, July 9–12, 2006.