A Canvass steganalyzer for double-compressed JPEG images

Pooja S. Paranjape
Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/etd

Part of the Electrical and Computer Engineering Commons

Recommended Citation

This Thesis is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
A Canvass steganalyzer for double-compressed JPEG images

by

Pooja Suhas Paranjape

A thesis submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE

Major: Electrical Engineering

Program of Study Committee:
Jennifer L. Davidson, Major Professor
Thomas Daniels
Namrata Vaswani

Iowa State University
Ames, Iowa
2011

Copyright © Pooja Suhas Paranjape, 2011. All rights reserved.
TABLE OF CONTENTS

LIST OF TABLES ........................................ iv

LIST OF FIGURES ..................................... v

ACKNOWLEDGEMENTS .................................. vii

ABSTRACT ............................................. viii

CHAPTER 1. INTRODUCTION ............................. 1

1.1 Background ...................................... 1

1.2 JPEG Compression ............................... 3

1.3 Double-compression ............................. 5

1.3.1 Primary Quality Factor = Secondary Quality Factor ........................ 7

1.3.2 Primary Quality Factor < Secondary Quality Factor ........................ 7

1.3.3 Primary Quality Factor > Secondary Quality Factor ........................ 7

1.4 Partially Ordered Markov Models: ................ 10

1.5 Pattern Classifier ............................... 13

1.5.1 Data preprocessing ........................... 14

1.5.2 Kernel selection ............................... 14

1.5.3 Cross-validation and Grid search .................. 15

1.6 Steganographic Algorithms ....................... 15

1.6.1 F5 ........................................... 15

1.6.2 Jsteg ........................................ 16

1.6.3 JP hide & seek ............................... 16

1.6.4 Outguess ..................................... 16

1.6.5 Steghide ...................................... 17
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Detection accuracy of Canvass when tested on double-compressed JPEG images <em>Note: for outguess 0.2 bpnz is used instead of 0.4 bpnz</em></th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 4.1</td>
<td>False positive rate of double-compression detector for (a) SQF= 75 and (b) SQF=80</td>
<td>40</td>
</tr>
<tr>
<td>Table 4.2</td>
<td>Average accuracy of DC Detectors for double-compressed images</td>
<td>41</td>
</tr>
<tr>
<td>Table 4.3</td>
<td>Comparison of false positive rates of POMM based detector and state-of-the-art detectors for (a) SQF=75 and (b) SQF=80</td>
<td>44</td>
</tr>
<tr>
<td>Table 4.4</td>
<td>Average detection accuracies of Pevny, Chen and POMM DC Detectors for (a) SQF = 75 and (b) SQF = 80</td>
<td>45</td>
</tr>
<tr>
<td>Table 4.5</td>
<td>Confusion matrices for cover Vs. stego detection for PQF &gt; SQF and (a) SQF = 75 and (b) SQF = 80</td>
<td>51</td>
</tr>
<tr>
<td>Table 4.6</td>
<td>Overall detection accuracy of (a) previous Canvass (b) our detector tested on double-compressed images</td>
<td>53</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 1.1 Proposed contribution to expand Canvass steganalyzer .............................................. 3
Figure 1.2 JPEG compression ........................................................................................................... 4
Figure 1.3 Double-compression in JPEG image ................................................................................. 6
Figure 1.4 Histograms of quantized DCT coefficients for mode (2, 2) in (a) single-compressed image, (b) double-compressed image with PQF < SQF ........................................ 8
Figure 1.5 Example of double-compressed coefficient values and resulting zero patterns at mode (2,2) ..................................................................................................................... 8
Figure 1.6 Histograms of quantized DCT coefficients for mode (2,2) in (a) single-compressed image (b) double-compressed image with PQF > SQF ........................................ 9
Figure 1.7 Example of double-compressed coefficient values and resulting peak patterns at mode (2,2). Bins 3 and 4 (row 1) combine to produce bin 3 (row 3) ........................................................................... 10
Figure 1.8 Example of acyclic digraph ............................................................................................ 11
Figure 1.9 Example of (a) cone and (b) adjacent lower neighbors of vertex $V_1$ .......................... 13
Figure 3.1 Proposed steganalyzer .................................................................................................... 23
Figure 3.2 Example of $f$ - $Y$ digraph and corresponding POMM Note: 1. $P(Y^h_k | f(Y^h_k))$ measures the frequency of occurrence of pair (3, 4) given the difference value of -1. 2. $P(*) | (*)$ defines the POMM associated with this horizontal $f$-$Y$ model ......................................................................................................................... 27
Figure 3.3 Horizontal difference array used in 2 different ways to create Chen’s features and POMM features .................................................................................................................. 27
Figure 3.4  Q-DCT arrays (a) Global array for intrablock POMMs (b) Mode array for interblock POMMs .............................. 30

Figure 4.1  Database used for training double-compression detector ......................... 38

Figure 4.2  Accuracy of double-compression detectors on double-compressed cover [(a),(b)], f5 [(c),(d)] and outguess [(e),(f)] images ............... 39

Figure 4.3  Comparison of accuracy of double-compression detectors for double-compressed Cover images ................................. 41

Figure 4.4  Comparison of accuracy of double-compression detectors for double-compressed f5 images for embedding levels 0.05 bpnz [(a),(b)], 0.1 bpnz [(c),(d)] and 0.4 bpnz [(e),(f)] .................... 42

Figure 4.5  Comparison of accuracy of double-compression detectors for double-compressed outguess images for embedding levels 0.05 bpnz [(a),(b)], 0.1 bpnz [(c),(d)] and 0.2 bpnz [(e),(f)] ...................... 43

Figure 4.6  Database used for training PQF splitter ................................. 45

Figure 4.7  Accuracy of PQF Splitters on double-compressed Cover, F5 and Outguess images ........................................ 47

Figure 4.8  Accuracy of cover Vs stego detector for PQF < SQF ....................... 48

Figure 4.9  Comparison of three detectors for cover Vs. stego detection for PQF > SQF, (SQF = 75) ............................................ 50

Figure 4.10  Accuracy of cover Vs stego detector for PQF > SQF ....................... 52

Figure 4.11  Accuracy of PQF detector ........................................... 54
ACKNOWLEDGEMENTS

It gives me a great pleasure to thank my advisor Dr. Jennifer Davidson for her support and encouragement during this research. She has been a great mentor throughout my graduate studies. I appreciate the time she has spent and all the valuable suggestions she has given for writing this thesis.

I thank Mr. Nick Multari of Boeing for funding this research and for the valuable feedbacks on this work. I would also like to thank my committee members Dr. Thomas Daniels and Dr. Namrata Vaswani for their support.

I am deeply indebted to my parents for their immense love, care and trust. I also thank Mr. Shirish Apte for the motivation and support which enabled me to pursue graduate studies. I extend my sincere thanks to my friends Trishna and Suvarna for always standing by my side during the thesis writing.
ABSTRACT

Steganography is the practice of hiding a secret message in innocent objects such that the very existence of the message is undetectable. Steganalysis, on the other hand, deals with finding the presence of such hidden messages. Canvass is software developed in [1] to perform JPEG image steganalysis. This software uses pattern recognizer to classify unknown images into cover (innocent) or stego (containing hidden message). The pattern recognizer, a support vector machine, is trained using the underlying statistical information in the cover and stego images. Some of the popular steganographic algorithms produce double-compressed JPEG images. A blind steganalyzer built on the assumption that it will see only single-compressed images gives misleading results of classification for such images. The goal of the current work is to develop a double-compression detector for JPEG images that extends the existing Canvass software. We develop a double-compression detector based on Partially Ordered Markov Models (POMMs) that acts as a pre-classifier to the blind steganalyzer, that in combination with patterns of relative histogram values of the quantized DCT coefficients, improves accuracy of detection. After detecting the double-compression, we carry out cover Vs stego detection and primary quality factor estimation on the double-compressed images. We compare our double-compression detector with two other state-of-the-art detectors. Our detector is found to have better performance compared to the state-of-the-art detectors. The current work considers a limited set of quality factors for double-compression but this novel method for steganalysis of double-compressed data is general and could be generalized for any combination of primary and secondary quality factors.
CHAPTER 1. INTRODUCTION

1.1 Background

Steganography is the practice of hiding a secret message or a payload in innocent objects such that the very existence of the message is undetectable. The goal of steganography is to embed a secret payload in a cover object in such a way that nobody apart from the sender and the receiver can detect the presence of the payload.

Steganalysis, on the other hand, deals with finding the presence of such hidden message. Steganalysis can be categorized as passive or active. Passive steganalysis deals with detecting the presence of embedded message. Active steganalysis seeks further information about the secret message such as length, embedding algorithm used and actual content. Steganalysis can also be classified as blind or targeted. In blind steganalysis, a single attack can be used for many steganographic algorithms. Generally, this is achieved by statistical modeling of the input image to convert it into lower dimensional features. In targeted steganalysis, a separate scheme is designed for each embedding algorithm. This allows one to use embedding signatures specific to each steganographic algorithm.

Multimedia objects such as digital images and videos are some of the popular choices for hiding a payload. This is because these objects are easily available and one can embed a reasonable amount of payload in them using various embedding algorithms. Amongst various formats of digital images such as BMP, PNG, JPEG, TIFF and so on, JPEG is the most commonly used format in the digital cameras as well as on the web. Also, since JPEG is a compressed file format, it requires lower bandwidth for transmission and lesser space for storage. Many steganographic embedding softwares for JPEG such as Outguess, F5 and Jsteg are freely available on the Internet [2]. This makes JPEG a good medium for steganography.
Hence, we focus our attention on the steganalysis of JPEG images.

If a JPEG image is compressed twice, each time using a different quantization matrix, then it is said to be double-compressed. Steganographic algorithms like F5 [3] and Outguess [4] can produce such double-compressed images during the process of embedding the payload. Existence of double-compression thus can suggest manipulation of the original image. Blind steganalyzers built on the assumption of single-compressed images give misleading results for the double-compressed images. As an example of this, we used the software Canvass to classify double-compressed images. Canvass [1] is a steganalysis software created by J. Jalan and J. Davidson to detect only single-compressed images. The detection accuracies of Canvass on double-compressed images are presented in Table 1.1. We use 117000 double-compressed cover, F5 and outguess images for testing. Canvass has a very high false positive rate of 54.61%. Also, the detection accuracies for F5 and outguess are poor. Thus, it is important to detect the presence of double-compression for satisfactory performance of the steganalytic algorithms. Detection of double-compression is a binary classification problem where a given image can be classified as either single-compressed or double-compressed. A double-compression detector can be thought of as a pre-classifier to multi-classifiers which detect a number of different steganographic methods used for embedding the payload.

<table>
<thead>
<tr>
<th>SQF = 75</th>
<th>% Detection Accuracy of Canvass</th>
<th>Average</th>
<th>FPR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cover</td>
<td>0.05 bpnz</td>
<td>45.39</td>
<td>45.39</td>
</tr>
<tr>
<td>F5</td>
<td>0.1 bpnz</td>
<td>28.84</td>
<td>34.54</td>
</tr>
<tr>
<td>Outguess</td>
<td>0.4 bpnz</td>
<td>42.04</td>
<td>57.73</td>
</tr>
</tbody>
</table>

Table 1.1 Detection accuracy of Canvass when tested on double-compressed JPEG images

Note: for outguess 0.2 bpnz is used instead of 0.4 bpnz

In Figure 1.1, we depict the general problem description. Our main goal is to expand the previous steganalyzer software package Canvass to include double-compressed images. In developing a double-compression detector, we want a model that has proven successful. We chose Pevny’s model [6]. In pursuing implementing that model, we encountered computational resource issues. We simply did not have the resources to create all the pattern classifiers. We
developed a different approach that accomplished the double-compression detection but that was significantly less computationally intensive.

In section 1.2, we first briefly discuss working of JPEG compression. In section 1.3, we discuss the double-compression phenomenon in detail, following which we present basic theory behind our proposed features called Partially Ordered Markov Models (POMMs) in section 1.4. The pattern classifier based on Support Vector Machines (SVM) is discussed section 1.5. We also present the description of different steganographic algorithms in section 1.6 and show how some of them produce double-compressed images.

1.2 JPEG Compression

JPEG stands for Joint Photographic Experts Group. It is a lossy compression algorithm that stores the data in the form of quantized DCT coefficients. Figure 1.2 shows the steps involved in JPEG compression.

An uncompressed image can be represented as a 2-dimensional array of pixels. Consider
such an image $A$ of size $M \times N$. First the uncompressed image in the spatial domain is divided into $8 \times 8$ non overlapping blocks. The value $a_{xy}$ represents a pixel value at location $(x,y)$ in an $8 \times 8$ block. These blocks are then transformed into the frequency domain using the Discrete Cosine Transform (DCT). The use of the DCT makes it possible to discard the high frequency information without affecting the low frequency information in the image by zeroing out most high frequencies. This process also preserves important visual information. The transformed coefficients are given by

$$b_{pq} = \alpha_p \alpha_q \sum_{x,y=0}^{7} a_{xy} \cos \left( \frac{(2x+1)\pi q}{16} \right) \cos \left( \frac{(2y+1)\pi p}{16} \right)$$ (1.1)

where

$$0 \leq p, q \leq 7$$

and

$$\alpha_p = \alpha_q = \begin{cases} 
\frac{1}{\sqrt{8}}, & \text{if } p, q = 0 \\
\frac{1}{2}, & \text{otherwise}
\end{cases}$$

The DCT is an invertible transform and the inverse DCT can be obtained as follows:

$$a_{xy} = \alpha_p \alpha_q \sum_{p,q=0}^{7} b_{pq} \cos \left( \frac{(2x+1)\pi p}{16} \right) \cos \left( \frac{(2y+1)\pi q}{16} \right)$$ (1.2)

where

$$0 \leq x, y \leq 7$$ and $\alpha_p$ and $\alpha_q$ are as above
The DCT coefficients $b_{pq}$ are then quantized by dividing each value point-wise using a $8 \times 8$ matrix $Q$, followed by rounding to the nearest integer. $Q$ is called the quantization matrix (QM). The quantized DCT coefficients are given by

$$B_{pq} = \left\lfloor \frac{b_{pq}}{Q_{pq}} \right\rfloor$$  \hspace{1cm} (1.3)

where $\lfloor \cdot \rfloor$ denotes rounding to the nearest integer and $0 \leq p, q \leq 7$.

Each location $(p,q)$ in the $8 \times 8$ block in the frequency domain has a unique two dimensional basis image associated with it, or 2-D frequency. We call this 2-D frequency a mode. The coefficient $b_{00}$ at mode $(0,0)$ is called the DC coefficient and the remaining coefficients are called AC coefficients.

Quantization is an irreversible lossy step as long as $Q_{pq} \neq 1$. Note that division by larger quantization steps results in more compressed data. The JPEG standard allows 100 different quality factors for compression. A quality factor is an integer between 1 and 100, inclusive. The quality factor 100 corresponds to the quantization matrix $Q_{pq} = 1$ for $0 \leq p, q \leq 7$, resulting in an uncompressed and highest possible quality image. Each quality factor is associated with a unique quantization matrix.

The quantized DCT coefficients are then Huffman encoded for further compression of data. This step is lossless.

### 1.3 Double-compression

In this section, we study the effect of double-compression on the DCT coefficients of the JPEG image. A JPEG image is double-compressed if it is compressed twice with different quantization matrices $Q^1$ and $Q^2$. In that case we have

$$B_{pq} = \left\lfloor \frac{b_{pq}}{Q^1_{pq}} \cdot \frac{Q^2_{pq}}{Q^2_{pq}} \right\rfloor$$  \hspace{1cm} (1.4)
Here, $Q^1$ is called the primary quantization matrix (PQM) and $Q^2$ is called the secondary quantization matrix (SQM). Figure 1.3 shows the double-compression process for a JPEG image. In this example, the image is first compressed at a lower quality factor QF = 50, using $Q^1$ as given on the left side of Figure 1.3. Next the image is uncompressed and then recompressed at a higher quality factor QF = 75, using $Q^2$ as shown on the right side of Figure 1.3. Note that a larger quality factor results in a quantization matrix with smaller values. We observe that, depending on the values of the quantization steps in primary and secondary quantization matrices, the histograms of the DCT coefficients exhibit different characteristic patterns. These patterns are discussed in sections 1.3.1 to 1.3.3 and are key to our efficient detectors.

We next discuss the three cases of relative quality factors for $Q^1$ and $Q^2$, providing analysis of the different patterns and how we used this information for our detectors.
1.3.1 Primary Quality Factor = Secondary Quality Factor

When the Primary Quality Factor (PQF) and Secondary Quality Factor (SQF) are identical, the quantized DCT coefficients have not changed in value. The mode histograms of the single and double-compressed images are identical in this case. The histogram of absolute values of quantized DCT coefficients of such an image is shown in Figure 1.4(a).

1.3.2 Primary Quality Factor < Secondary Quality Factor

When the Primary Quality Factor (PQF) is smaller than the Secondary Quality Factor (SQF), the image is first coarsely quantized and then finely quantized such as given in the example in Figure 1.3. The quantized DCT coefficients can thus take values only from the set 0, n, 2n, 3n,... where n is determined from equation 1.4. The histogram exhibits zeros at remaining points. The histogram in Figure 1.4(b) shows the characteristic zero pattern for a double-compressed image in this case. Here the primary quality factor is 63 and secondary quality factor is 75. The primary quantization step for mode (2,2) is 9 whereas the secondary quantization step is 6. After the first compression step, the dequantized DCT coefficients can thus take values which are multiples of 9. After the second compression that includes requantization with step size of 6 and rounding, the quantized coefficients can take values only from the set 0, 2, 3, 4, 6, 8, 9, 11, .... We notice that these values are the rounded integer multiples of \( n = \frac{9}{6} = 1.5 \). As a result, zeros occur at locations 1, 4, 7, 10 and so on. This is summarized in Figure 1.5.

We exploit this characteristic to develop features to classify single Vs. double-compressed images as well as cover Vs. stego images.

1.3.3 Primary Quality Factor > Secondary Quality Factor

When the primary quality factor is greater than the secondary quality factor, the mode histograms of the DCT coefficients typically exhibit peaks at certain places. The locations of these peaks vary according to the combinations of primary and secondary quantization steps in a way similar to the previous case. By experimentation on the image data, we determine that
Figure 1.4  Histograms of quantized DCT coefficients for mode (2, 2) in (a) single-compressed image, (b) double-compressed image with PQF < SQF

| Quantized DCT coefficient values (Q_{12}^{12}=9) | \begin{barray}{|c|c|c|c|c|c|c|c|c|c|c|} \hline b_{12}^{12} + \frac{Q_{12}^{12}}{Q_{12}^{12}} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ \hline \end{barray} |
| De-quantized DCT coefficient values (integer multiples of step Q_{12}^{12}=9) | \begin{barray}{|c|c|c|c|c|c|c|c|c|c|c|} \hline b_{12}^{12} \times \frac{Q_{12}^{12}}{Q_{12}^{12}} & 0 & 9 & 18 & 27 & 36 & 45 & 54 & 63 & 72 & 81 & 90 \\ \hline \end{barray} |
| Re-quantized DCT coefficient values, quantization step Q_{12}^{12}=6 (rounded integer multiples of n=1.5) | \begin{barray}{|c|c|c|c|c|c|c|c|c|c|c|} \hline b_{12}^{12} \times \frac{Q_{12}^{12}}{Q_{12}^{12}} & 0 & 2 & 3 & 5 & 6 & 8 & 9 & 11 & 12 & 14 & 15 \\ \hline \end{barray} |

Figure 1.5  Example of double-compressed coefficient values and resulting zero patterns at mode (2,2)
if there is 20% increase in the histogram values (bin heights) followed by a drop in the values, it indicates a peak. The value 20% is chosen after a series of trials and errors. Equation 1.4 also models this phenomenon.

In Figure 1.6(a) and 1.6(b), we present histograms to demonstrate this phenomenon. In this case, the Primary Quality Factor is 80 and the Secondary Quality Factor is 75. The primary quantization step for mode (2,2) is 5 and the secondary quantization step is 6. The dequantized coefficients after the first compression are thus integer multiples of 5. After requantization with step 6, peaks occur at locations 3, 8 and 13 and so on. These are indicated by the arrows in Figure 1.6(b). Figure 1.7 explains this phenomenon in detail. A peak value occurs at bin value 3 after second compression because bin values 3 and 4 from the first compression end up in bin values 3 after the second compression.

While peak values created this way can appear visually in the histogram to vary little to none, we basically count the number of times that the peak-valley combination occurs in a mode histogram, and if it occurs “often enough”, we claim double-compression with PQF > SQF.
1.4 Partially Ordered Markov Models:

We propose to use Partially Ordered Markov Models (POMMs) [7] as features for double-compression detection and steganalysis of single and double-compressed images. The idea is to represent an image in lower dimensions to train a pattern classifier. The novel contribution of this approach is that while POMMs have been used for cover vs. stego detection in single-compressed image at 75 quality factor, they had not been used for double-compression detection or for a binary PQF detector. A detailed description of use of features for classification problems is given in section 3.3. In this section, we discuss the basic theory and definition of POMMs.

POMMs are a subclass of Markov Random Fields (MRFs) which have been used successfully for stochastic texture synthesis as well as texture modeling. However, MRFs are complex to use for solving problems related to pattern recognition such as parameter estimation or texture classification. This is because of the problem in specifying an explicit joint probability that is computationally feasible. Markov Mesh Models (MMMs) are a subclass of MRFs and were introduced by Abend et al. [8]. Under minimal reasonable assumptions, MMMs allow one to
specify an explicit closed form of the joint probability of the random variables (r.v.s) at hand. The joint probabilities are expressed in terms of conditional probabilities. Unlike an undirected neighborhood used by the MRFs, the MMMs use a directional neighborhood to express the conditional probabilities. It was also shown in [8] that the conditional probability of one r.v. given the rest of the r.v.s can be expressed in terms of r.v.s in a local spatial neighborhood. Due to their directional or causal nature, the MMMs are not used extensively in image analysis areas except image restoration and segmentation. However, there is an underlying notion of partial ordering of the pixel locations in the definition of MMMs. A partial ordering is an order where every pair of elements is not necessarily related. This notion of partial ordering allows the generalization of MMMs into POMMs. MMMs are used for data that are arranged on a rectangular lattice array but POMMs are applicable to data arranged on any type of array such as rectangular, hexagonal and n-dimensional arrays, or randomly arranged spatially.

Before discussing the POMMs, we present necessary definitions for a finite acyclic digraph $(V, E)$ with corresponding poset $(V, \prec)$. Here $V = (V_1, V_2, \ldots, V_k)$ is the set of vertices and $E$ is the set of directed edges given by $E = \{(i, j) : V_i, V_j \in V \text{ and } (i, j) \text{ is an edge with tail on } i \text{ and head on } j\}$. $(V, E)$ is called acyclic when there does not exist a set of edges $(i_1, j_1), (i_2, j_2), \ldots, (i_k, j_k)$ where $j_n = i_{n+1}, n = 1, 2, \ldots, k - 1$ and $j_k = i_1$. Figure 1.8 shows an acyclic digraph. We notice that there is no path of directed edges that start and end at the same vertex.

**Definition 1:** A set of elements $V$ with a binary relation $\prec$ is said to have a partial order with respect to $\prec$ if:

1. $a \prec a, \forall a \in V$ (reflexivity)
2. $a \prec b, b \prec c \Rightarrow a \prec c$ (transitivity)

3. If $a \prec b$ and $b \prec a$, then $a = b$ (anti-symmetry)

In this case, $(V, \prec)$ is called a partially ordered set or a poset. We now review the correspondence between a finite poset $(V, \prec)$ and an acyclic directed graph $(V, E)$. For a given acyclic digraph, we can define a binary relation $\prec$ by using the existing edge set which induces the partial order $\prec$ in the natural way. On the other hand, for a given finite poset, there is a set of digraphs with different edge sets that have the same underlying poset $(V, \prec)$. Thus there exists a many-to-one correspondence between one poset and many acyclic digraphs. The different edge sets lead to the neighborhood relationship between pixel values. These relationships are used to describe the conditional probabilities on neighborhood pixels for the probability model. The probability model for the image at hand is constructed using the acyclic digraph that the image analyst chooses, since the many-to-one relationship between one poset and many acyclic directed graphs does not allow a unique probability model to be described.

**Definition 2:** For $V_i, V_j \in (V, \prec)$, $V_i$ is covered by $V_j$ if $V_i \prec V_j$ and $V_i \prec V_k \prec V_j$ for no $k$.

**Definition 3:** For any $B \in V$, the cone of $B$ is the set $cone_B = \{C \in V : C \prec B, C \neq B\}$

**Definition 4:** For any $B \in V$, the adjacent lower neighbors of $B$ are elements in $C$ such that $(C, B)$ is a directed edge in $(V, E)$. Formally, $adj_\prec B = \{C : (C, B) \text{ is a directed edge in } (V, E)\}$. Note that $adj_\prec B \subseteq cone_B$.

**Definition 5:** Any $B \in V$ is a minimal element if there is no element $C \in V$ such that $C \prec B$, that is, there is no directed edge $(C, B)$ into $B$. Let $L^0$ be the set of minimal elements in the poset.

Figure 1.9 explains these definitions pictorially.

With this background, we now proceed towards defining POMMs. Let $P(A)$ be the discrete probability for r.v. $A$ and $P(A|B)$ be the conditional probability of r.v $A$ given another r.v. $B$.

**Definition 6:** Consider a finite acyclic digraph $(V, E)$ of r.v.s with the corresponding poset $(V, \prec)$. For $B \in V$, consider a set $Y_B$ of r.v.s not related to $B$, $Y_B = \{C : B \text{ and } C \text{ are not related}\}$. Then $(V, \prec)$ is called a partially ordered Markov model (POMM) if for any $B \in V \setminus L^0$...
Figure 1.9 Example of (a) cone and (b) adjacent lower neighbors of vertex $V_1$

and any subset $U_B \subset Y_B$ we have

$$P(B \mid coneB, U_B) = P(B \mid adj_{\prec}B)$$  \hspace{1cm} (1.5)

In this case, the lower adjacent neighbors describe the Markovian property of the model. In other words, the probability of B depends only on those neighbors $\prec B$ that are connected by an edge.

### 1.5 Pattern Classifier

Detection of double-compression is a binary classification problem wherein an input image is classified as single-compressed or double-compressed. We propose to use a blind steganalyzer to solve this classification problem. Generally, blind steganalysis is carried out by converting the input image to lower dimensional feature space. These features are then used to train a pattern classifier. Amongst various pattern classification techniques, Support Vector Machines (SVMs) prove to be very powerful for the binary classification. SVMs use either linear or kernel-based supervised machine learning algorithms. Linear SVMs are seldom suitable for real world data that is hard to separate without error by a hyperplane. In the following paragraph, we introduce the basic theory behind SVMs. More details can be found in [9, 10].

Consider a given set of training pairs $(\vec{x}_i, y_i), i = 1, 2, ..., l$, where $\vec{x}_i \in R^n$, $n$ represents the number of features, and $y \in \{-1, 1\}$, represents the class. The binary SVMs require solution
in the form of a weight vector $\mathbf{w}$ to the following optimization problem:

$$\min_{\mathbf{w}, b, \xi} \left( \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^{l} \xi_i \right)$$

subject to $y_i (\mathbf{w}^T \Phi(\mathbf{x}_i) + b) \geq 1 - \xi_i,
\xi_i \geq 0$

The training data are mapped into a higher dimensional space by function $\Phi$. Then a linear separating hyperplane is found with the maximal margin of space between the hyperplane and data in the higher dimensional space. In this case, $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$ is called the kernel function whereas $C > 0$ is the penalty parameter of the error term $\xi$.

There are two distinct phases involved while using a SVM for classification. In the first phase, the SVM is trained on the features extracted from a large number of images, that is, a weight vector $\mathbf{w}$ is found satisfying equation 1.5. In the second phase, the SVM is tested on the data which is previously unseen, that is, a vector of features from an image with unknown class is passed through the SVM and its class is output. We now discuss different steps involved in training the SVM.

### 1.5.1 Data preprocessing

First the input data is linearly scaled so that all the elements are in the range $[-1, 1]$ or $[0, 1]$. The same scaling is used for both the training and the testing data. This step is necessary to prevent large feature values from dominating small values.

### 1.5.2 Kernel selection

We have various options for the kernel functions such as Gaussian, Radial Basis Function (RBF), Polynomial, Sigmoid etc. We use RBF kernel for all the experiments. RBF kernel is most suitable when the number of features is small and the data needs to be mapped to higher
dimensional space using a non linear kernel. Also, RBF kernel is numerically less complex than polynomial and sigmoid kernels. It is given by:

\[ K(x_i, x_j) = \exp(-\gamma\|x_i - x_j\|^2), \gamma > 0 \quad (1.7) \]

where \( \gamma \) is the kernel parameter and the norm used is the Euclidean distance.

1.5.3 Cross-validation and Grid search

The SVM library that we used [9] determines the optimum values for the kernel parameters \( C \) and \( \gamma \). This is done by performing an exhaustive grid search on the predefined points. Cross validation is required to ensure good performance of the detector on the unknown data. In \( v \)-fold cross-validation, the training data is divided into \( v \) subsets of equal sample size and \( v-1 \) SVMs are created. Each SVM uses \( v-1 \) subsets for training and the remaining subset is used as unknown test data. This gives the estimate of prediction accuracy for the unknown data by averaging the accuracy values. Cross validation can also help prevent over-fitting the data to the solution, resulting in better test accuracy [9]. In our experiments we use \( v = 5 \).

1.6 Steganographic Algorithms

In this section, we present an overview of five popular steganographic algorithms which are considered in the experimental set up.

1.6.1 F5

F5 was introduced by A. Westfeld [3]. The algorithm uses matrix embedding to embed the payload bits into randomly chosen sites in the DCT coefficients array. Instead of LSB replacement, the DCT coefficient value is decremented by one if the message bit is different than the the LSB. This scheme is robust to the \( \chi^2 \) attack because of minimum number of changes in the cover histogram. F5 can embed data that is up to 13% of the size of the cover image.
If the input image is in uncompressed format, F5 compresses it using a default quantization matrix. If the input image is already in JPEG format, then compression with the default quantization matrix can lead to the double-compressed JPEG image. The default quality factor for F5 is 80.

1.6.2 Jsteg

Jsteg was proposed by Derek Upham [11] and it was the first publicly available embedding algorithm for JPEG images. Jsteg uses LSB embedding technique. A secret payload is embedded by sequentially replacing the LSBs of the quantized DCT coefficients. The zero-valued and one-valued coefficients are not used for embedding. This is because these coefficients are large in number and embedding in them leads to perceptible artifacts in the histograms. It is a popular embedding algorithm due to its high embedding capacity. The maximum size of the payload that can be embedded is approximately 12% of the cover image size.

1.6.3 JP hide & seek

JP hide & seek (JPHS) designed by Allan Latham [12] is another LSB replacement algorithm. The DCT coefficients used for embedding are selected based on a fixed table. The large valued coefficients are used first. A pseudo random number generator controlled by a secret key is used to decide random skipping of the coefficients. JP hide & seek also uses second LSB for embedding if the payload size is larger than the number of LSBs available for embedding.

1.6.4 Outguess

Outguess was proposed by N. Provos [13]. Outguess first embeds the data into the quantized DCT coefficients. The second step makes corrections to the statistical changes involved in the first step. This is done by changing the coefficients that were not involved in the embedding process earlier. This preserves the global histograms and makes the scheme robust to the \( \chi^2 \) attack. The maximum size of payload is 6% of the cover image size.

The default quality factor of outguess is 75. When a JPEG image is input to the outguess algorithm, it is first decompressed and re-compressed at quality factor 75 and then the payload
is embedded into the image. This process results in double-compressed JPEG images.

1.6.5 Steghide

Steghide is based on the graph theoretic approach. A compressed and encrypted payload is embedded into the cover image by random site visitation. If the LSB at the visited site matches the payload bit, nothing is done. If there is a mismatch between the two bits, then a different location is searched graph-theoretically such that the cover bit matches the payload bit. The values at these two sites are then switched. This is done for all such possible pairs.
CHAPTER 2. REVIEW OF LITERATURE

Double-compression in JPEG images almost always indicates image manipulation. An innocent image with a certain quality factor can sometimes be saved at a different default quality factor, resulting in double-compression of an innocent image. In image forensics, double-compression is used to detect image forgery. A doctored image may be created by pasting a small part of one image onto another image. If the quality factors of these two images are different, it results in double-compression of the pasted region.

In [14] Fridrich et al. proposed a targeted attack for JPEG images embedded using F5 steganographic algorithm. The method mainly considered single-compressed images and was based on estimating the cover histogram from the stego image and determining the relative number of modifications in the histogram values. The authors also suggested a method to estimate the primary quantization steps based on simulating the double-compression using all the candidate quantization matrices. This method was also applied for attacking the outguess steganographic algorithm in [15]. However, in both [14] and [15], the authors assumed that the images are already known to be double-compressed. No method was implemented for the double-compression detection.

Lukas and Fridrich extended this idea in [16] and proposed three methods for detection of primary quantization steps in double-compressed JPEG images. Two of these methods were based on simulated double-compression as in [14]. The methods were computationally intensive and did not lead to reliable results. The third method was based on neural networks and it outperformed the first two methods. However, in the latter paper the authors assumed that the double-compressed images were cover images only.

He et al. [17] proposed a scheme to detect doctored JPEG images for forensic applications based on double quantization effects seen in the DCT coefficient histograms. The method used
a Bayesian approach for detecting doctored blocks in the images. A support vector machine was trained on the four dimensional features that are similar to Fisher discriminator in pattern recognition. This method is not efficient when a high quality image is re-compressed using a lower quality factor. Also, for more accurate formulation of the feature vectors, the method needs the primary quality factor which cannot be estimated from the proposed algorithm. In [18], M. Sorell developed a method to detect the primary quantization coefficients of double-compressed images for forensic purposes. The method was used to detect re-quantization in innocent JPEG images to identify the source of the images. Methods for detecting doctoring of JPEG images, where a region undergoes double-compression after being cut and pasted into a larger image, have been investigated in [17, 19, 18]. Although some of these methods lead to accurate detection of double-compression, the application was for forensic purposes, not steganalysis. Since the embedding process for steganography can change the image statistics significantly, application of these methods for steganalysis might involve significant algorithmic modifications.

In [20], T. Pevny and J. Fridrich proposed a stego detector for both single and double-compressed images. They used a neural network-based primary quantization step detector to detect double-compression. Depending on the result of this detector, the image was sent to one of the two banks of multi-classifiers designed for single and double-compressed images. The detection of double-compression was based on the naive approach of comparing the primary and secondary quantization steps and it led to inaccuracies in the detection. To overcome this, a separate double-compression detector was designed in [21] and it was used as a pre-classifier to the stego detector. Instead of neural networks, the authors used Support Vector Machines. The SVMs were trained on features based on first order statistics of the mode histograms of quantized DCT coefficients. The primary quantization step detection followed the double-compression detection. For each possible combination of primary and secondary quantization steps, the authors used a separate SVM which led to a total of 159 SVMs. This approach, while increasing accuracies, is very compute-intensive.

C. Chen et al. proposed similar machine learning based scheme for JPEG double-compression detection [22] similar to [20]. The proposed 324 dimensional features were modeled using a
Markov process and transition probability matrices of 2D difference arrays of quantized DCT coefficients were used to create the feature vectors. Then a support vector machine was trained on these features. The method is discussed in detail in Chapter 3 and the results are compared with our proposed detector in Chapter 4.

Pevny et al. created a complete multi-classifier for single and double-compressed images [6] by combining all the modules from [20, 21]. The double-compression detector was based on histogram mode statistics. The primary quality step estimation was done by a bank of SVMs designed for all the possible combinations of primary and secondary quality steps. Then two banks of multi-classifiers were created for single and double-compressed images. This model is also highly compute-intensive and is discussed in detail in Chapter 3 and the performance is compared with our proposed detector in Chapter 4.
CHAPTER 3. PROPOSED APPROACH

Canvass is a steganalysis software developed by J. Davidson and J. Jalan [1]. The software can successfully detect if a given image is cover or stego and assign the stego image to one of the five classes: f5, jp hide & seek, jsteg, outguess or steghide. The software works for single-compressed images with size larger than $512 \times 512$ pixels and quality factor of 75. The same idea can be extended to create steganalyzers for images with different sizes and quality factors, all single-compressed.

Canvass uses a pattern classifier trained only on single-compressed images and has poor detection accuracy for double-compressed images. This is because of the statistical artifacts introduced by the double-compression process. It is observed that many double-compressed cover images, when passed through Canvass, get misclassified as stego images. The detection accuracy for double-compressed f5 and outguess images is also very poor. The high false positive rate and low true positive rate of Canvass for double-compressed images make it necessary to design a separate scheme to handle double-compressed images.

The model proposed by T. Pevny et al. [6] proved successful for designing multi-classifiers for both single and double-compressed images. The main disadvantage we see is very heavy computational resource requirements. The multi-classifier requires 15 SVMs for stego Vs. cover detection of single-compressed images. 159 additional SVMs are required for primary quality steps estimation whereas 3 SVMs are required for stego Vs. cover detection of double-compressed images. Considering the time required for training and testing of each of these SVMs and the computational resources at hand, we decided to create a different approach that would minimize the number of SVMs required, thus saving a lot of computational time and being realistically implementable on our 1, 3 years old computer.

A major contribution of our approach towards time saving is achieved in the primary quality
factor estimation step. Instead of using SVM for each possible combination of primary and secondary quality step, we use an analytical approach based on the signature patterns in the histograms. The analytical method does not involve training and testing of SVM and is significantly less computationally intensive.

The goal of this research is to create a steganalyzer that can:

1. Classify an image as single-compressed or double-compressed (Box 1).

2. Classify a double-compressed image as cover or stego and estimate the primary quality factor from a set of standard JPEG quality factors for further analysis of the image using targeted attacks. [14, 15]. (Box 2).

3. Classify a single-compressed image as cover or stego (Box 3). (Note: Only QF = 75 case is implemented in the current Canvass software.)

We assume three classes for double-compressed images: cover, f5 and outguess. In practice, double-compressed images can have any quality factor but we restrict the scope to secondary quality factors 75 and 80 which are the default quality factor for outguess and f5 respectively.

In section 3.1, we propose the overall scheme for steganalysis of single and double-compressed images. In section 3.2, we describe the image database used for training and testing various SVMs. Section 3.3 gives a detailed description of the features used for steganalysis whereas section 3.4 contains a brief discussion about features used by our state-of-the-art double-compression detectors.

### 3.1 Proposed Steganalyzer

Figure 3.1 shows the complete flow of the proposed steganalyzer. In the following sections, we describe the building blocks of the steganalyzer. If the observed quality factor of the input image is 75 or 80, there is a high probability of double-compression. This image is passed to the double-compression detector.

If the image is single-compressed at QF = 75, it is passed to the existing Canvass. If the image has QF ≠ 75, ≠ 80, then we do not process it at this time.
Figure 3.1 Proposed steganalyzer

If the image is found to be double-compressed, it is further given to PQF splitter module. The PQF splitter determines if the PQF of the image is less than or greater than its SQF and accordingly classifies the image into these two classes. This step helps in the further processing because it is observed that the signature patterns in the histograms depend on the relation between the PQF and the SQF. We solve the cover Vs. stego detection problem in 2 different ways depending on whether PQF < SQF or PQF > SQF.

When the PQF is less than the SQF, we use the analytical approach based on the histogram signature patterns to achieve stego Vs cover classification. When the PQF is greater than the SQF, we use a multi-class SVM-based pattern classifier.

All the modules shown in the steganalyzer block diagram in Figure 3.1 are explained in detail in Chapter 4.

3.2 Image Database

In this section, we describe the database used for all the experiments. This description is necessary because the performance of a SVM-based classifier depends on the database used for training.

We use images from BOWS-2 database available at [23]. This database was created for the BOWS-2 contest. It consists of approximately 10000 grey-scale images in pgm format. The images are of uniform size of 512 × 512 pixels.

The BOWS-2 database is divided into two mutually exclusive sets of equal size. One set is used for creating the training database and other set is used for creating the testing database.
This division allows use of test images which are previously not seen by the SVM. For all the
experiments, a total of 30000 images are used for training, 15000 of each class.

For single-compressed images, five steganographic algorithms are considered: f5, jsteg, jp
hide & seek, outguess and steghide. For double-compressed images, only f5 and outguess are
considered because we assume that these are the only algorithms that are likely to produce
double-compressed images. For all the embedding algorithms except outguess, three embedding
levels are used: 0.05 bpnz, 0.1 bpnz and 0.4 bpnz. Outguess fails to embed the payload at
0.4 bpnz. Hence 0.2 bpnz embedding rate is used instead. In this case, Bits Per Non-Zero
ac coefficient (bpnz) is used to describe the payload length. Each steganographic algorithm
embeds a binary payload consisting of a randomly generated bitstream into the quantized DCT
coefficients array of the JPEG image, where only non-zero and non-one ac coefficients are used.

In our experiments, we consider 33 primary quality factors from the set

\[ S = \{63, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89,
90, 91, 92, 93, 94\}. \]

Secondary quality factors are limited to 75 and 80. We ignore primary quality factors 64,
65, 66 and 67 because f5 and outguess fail to embed the payload in the images with these
quality factors. As a side-note, a possible future investigation could research if this property is
somehow helpful to the steganalysis problem.

### 3.3 Features

As discussed previously, targeted steganalysis calls for prior information about embedding
algorithms used and corresponding embedding signatures. We use blind steganalysis methods
which do not require such prior information and single scheme works for attacking a wide range
of embedding algorithms. Blind steganalysis generally uses a machine learning-based approach.
We use support vector machines to carry out the pattern classification task. All the pattern
classifiers in our experiments are binary, meaning they classify an input image into one of the
two classes. In order to train a pattern classifier, we need to represent the images from higher
dimensional space using features from lower dimensional space. These feature vectors are then
used during the SVM training phase to create the decision boundaries that separate the feature
space into two separate regions corresponding to the two classes under consideration. When an unknown test image is input to the trained classifier, a similar feature vector is extracted. Depending on the location of this vector in the feature space, the image is classified.

There is a wide variety of features that have been used for pattern classification. Farid [24] used features based on the linear prediction error of wavelet sub-band coefficients. In [25], features based on quantized DCT coefficients were used for blind steganalysis. Shi et al. [26] used a Markov process to model four directional differences between the neighboring quantized DCT coefficients to create features. In all the methods, the idea is to use features that are sensitive to the changes introduced by the embedding process. Such features contain the information that can effectively separate the classes under consideration.

We describe POMMs in section 1.3. We discuss how the POMMs exploit neighborhood dependencies amongst the quantized DCT coefficients. POMMs have been successfully used for the forward and inverse texture modeling problem. In order to improve the performance of the POMMs, we additionally use features based on the histogram signature patterns. In the following section, we discuss the use of POMMs and histogram-based features for steganalysis.

3.3.1 Partially Ordered Markov Models (POMMs) for steganalysis

We are interested in using POMMs for modeling steganographic changes to an image. For steganalysis applications, we create features that use directly the quantized DCT coefficient values that are modeled by a POMM. We describe a general approach that lets the steganalyst use her expertise to construct such a POMM. First, subsets of pixels are chosen that contain relative information that may have changed after embedding a payload. Next, a function \( f \) from the set of subsets \( V \) to the real numbers \( \mathbb{R} \) is found that is used to quantify the change in values that occur after embedding, and applied to each subset under consideration. Then, an acyclic directed graph is created, where \( V \) is the set of subsets and values in the range of \( f \) and \( E \) is the binary relation describing the function and its range. The induced acyclic digraph \( (V, E) \) is found and a POMM is created using this acyclic digraph. This gives rise to the conditional probabilities \( P(B \mid \text{adj} \prec B) \) which are used as features for steganalysis.

Let \( A \) be a \( M \times N \) array of quantized DCT coefficients: \( A = \{A_{i,j} : 1 \leq i \leq M, 1 \leq j \leq N\} \).
Let \( Y = \{Y_1, Y_2, ..., Y_t\} \) be a collection of subsets of r.v.s in A. For our purposes, we assume that \( Y \) is a shift invariant ordered set, although this could be generalized. For example, consider a case where \( Y^h_1 = \{A_{1,1}, A_{1,2}\}, Y^h_2 = \{A_{1,2}, A_{1,3}\} \), and so on. Each \( Y^h_i \) is a set containing two pixels that are adjacent in the horizontal direction. Let \( Y = Y^h = \{Y^h_1, Y^h_2, ..., Y^h_k\} \) contain all such sets of r.v.s in the array A. Define a function \( f: Y^h \to \mathbb{R} \), where \( \mathbb{R} \) is the set of real numbers, by

\[
 f(y_1, y_2) = y_1 - y_2 \\
 f(Y^h_i) = f(A_{j,k}, A_{j,k+1}) = A_{j,k} - A_{j,k+1}, \text{ for some indices } i, j \text{ and } k.
\]

In this case, \( f(Y_i) \) is the image of \( Y_i \) under \( f \) and \( Y_i \) is the pre-image of \( f(Y_i) \). An acyclic digraph \((V, E)\) is created from the set of vertices \( V = Y \cup f(Y) \) and the set of edges \( E = \{E_i\} \) between an element of the range from \( f \) and an element of \( Y \), where edge \( E_i = (f(Y_i), Y_i) \) has tail on image \( f(Y_i) \) and head on pre-image \( Y_i \). The pair \((V, E)\) forms a function-subset or \( f-Y \) acyclic digraph. This acyclic digraph is used to create a set of POMMs whose conditional probabilities are used as features. If \( f \) is a function which exploits the dependency amongst the quantized DCT coefficients, then it is considered useful for steganalysis. For such a function \( f \), the POMM defined by its conditional probabilities \( P(Y_k | f(Y_k)) \), is a measure of the frequency of occurrence of pre-image of \( f(Y_k) \), and can be used to distinguish between cover and stego images. This is the motivation for using \( f-Y \) acyclic digraph.

Figure 3.2 shows an example of \( f-Y \) directed graph for the horizontal differences \( Y = Y^h \). There are two subsets \( Y_{112} \) and \( Y_{249} \) shown which represent a portion of the quantized DCT coefficients array. Function \( f \), when applied to these subsets produces a value of -1. So the probability \( P(3, 4 | (3 - 4)) = P(3, 4 | -1) \) measures the frequency of occurrence of value -1 when it occurs from the pair \((3, 4)\). A collection of all such conditional probabilities are captured by the POMMs.

As mentioned above, function \( f = y_1 - y_2 \), where \( y_1 \) and \( y_2 \) are adjacent pixels, is a very useful feature for steganalysis. The Markov transition matrices from directional difference arrays created in this manner have been successfully used in [22], [26]. We use the function \( f \) to create a set of POMMs whose measures are not transition probabilities, but conditional
Figure 3.2  Example of $f - Y$ digraph and corresponding POMM Note: 1. $P(Y^h_k \mid f(Y^h_k))$ measures the frequency of occurrence of pair (3, 4) given the difference value of -1.
2. $P(\ast \mid \ast)$ defines the POMM associated with this horizontal $f-Y$ model.

$Y_{112} =$

-1: $r = f(a_{ij}, a_{i,j+1}) = a_{ij} - a_{i,j+1}$

= $f(a_{hk}, a_{i,k+1}) = a_{hk} - a_{i,k+1}$

Figure 3.3  Horizontal difference array used in 2 different ways to create Chen’s features and POMM features.

Chen: $P(a-b \mid b-c)$; POMM: $P(a,b \mid a-b)$

probabilities. In [22] and [26], the authors use the conditional probabilities $P(a-b \mid b-c)$ where a, b, and c are difference values created from adjacent pixels in the Q-DCT array. These are the Markov transition probabilities used by the authors as features. See Figure 3.3 for an example of the horizontal direction. In this context, the horizontal POMM is the conditional probability, which is transition probability.

Specifically, on a rectangular array $A$ of r.v.s we consider four directional subsets in horizontal, vertical, diagonal and minor diagonal directions given by $Y^h = (A_{i,k}, A_{i,k+1}), Y^v = (A_{i,k}, A_{i+1,k}), Y^d = (A_{i,k}, A_{i+1,k+1})$ and $Y^m = (A_{i+1,k}, A_{i,k+1})$. From the four sets $Y^h, Y^v, Y^d$
and \( Y^m \) thus obtained, we create four acyclic digraphs \((V^*, E^*)\) where \( V^* = (Y^* \cup f(Y^*)) \) and \( E^* = \{E^*_i : E^*_i = (f(Y^*_i), Y^*_i)\} \), and \( * \in \{h, v, d, m\} \). For each digraph, a POMM is defined by its conditional probability given by:

\[
P^*(Y^* \mid f(Y^*)) = \frac{P^*(Y^* \mid f(Y^*))}{P^*(f(Y^*))}
\]  

(3.1)

These probabilities are calculated from the histogram bins. The histograms are clipped (not thresholded) to values between \([-T, T]\). This not only avoids the sparse pdf values at the histogram ends but also reduces the number of computations and the size of the feature vector. Thus, we consider probabilities where \(-T \leq A_{i,j} \leq T\). This limits the values of \( P^* \) to \((2T + 1)^2\) in each direction. In our experiments, we use \( T = 3 \). We now define a matrix \( F^* \) of size \((2T + 1) \times (2T + 1)\) by

\[
F^*(w, z) = P^*(Y^* \mid f(Y^*)) = P^*(w, z \mid f(w - z)) = P^*(w, z \mid w - z)
\]

Finally, a set of \((2T + 1)^2\) features \( F(w, z) \) can then be defined as the average of \( F^*(w, z) \) over the four POMMs:

\[
F(w, z) = \frac{1}{4} \sum_{* \in \{h,v,d,m\}} F^*(w, z) = \frac{1}{4} \sum_{* \in \{h,v,d,m\}} P^*(Y^* \mid f(Y^*)) = \frac{1}{4} \sum_{* \in \{h,v,d,m\}} P^*(w, z \mid f(w - z))
\]  

(3.2)

Here, \(-T \leq w, z \leq T\). The POMMs are applied on the global array to capture intrablock dependencies and on the mode arrays to capture interblock dependencies. The next two sections describe these features in detail.

**3.3.1.1 Intrablock features**

The embedding process affects the correlation between quantized DCT coefficients in a single 8 \( \times \) 8 block. These changes are quantified by the POMMs defined on the global array. To capture the disturbances in the overall smoothness and continuity in the image, we use the values in the matrix \( F \) defined in equation 3.2 as part of our feature set. This gives \((2T + 1)^2\) feature values. Since these values describe dependencies among DCT coefficients within a block, we refer to these features as *intrablock* features.
3.3.1.2 Interblock features

The DCT coefficients located at the same positions or modes in the $8 \times 8$ blocks are collected to form *mode arrays*. Then these mode arrays are modeled using the POMMs. This captures the changes caused between blocks by the process of embedding. Frequencies in the spatial signals that cover more than an $8 \times 8$ block are represented in adjacent blocks in the DCT domain. For a $M \times N$ array $A$, the number of $8 \times 8$ blocks are given by $N_r \ast N_c$ where $N_r = \lceil \frac{M}{8} \rceil$ and $N_c = \lceil \frac{N}{8} \rceil$. For the 64 modes in $8 \times 8$ array, 64 mode arrays are created. The conditional probabilities are calculated in each mode array using equation 3.2. The final features are obtained by averaging the conditional probabilities over all 64 arrays. This gives additional $(2T + 1)^2$ interblock feature values.

Figure 3.4 shows the arrays of Q-DCT coefficients used to create the intrablock and interblock POMMs. Figure 3.4(a) shows the global array of $512 \times 512$ size on which the intrablock POMM features are defined. Figure 3.4(b) shows the mode array of $64 \times 64$ size created by collecting all quantized DCT coefficients at mode $(1, 1)$. There are 64 modes in each $8 \times 8$ block which result in 64 such mode arrays. The POMMs obtained from these mode arrays are averaged to get the final interblock feature values.

**Calibration:** To calibrate a JPEG image, it is decompressed to the spatial domain and then compressed back after cropping by a few pixels in both directions. The idea is to get rid of “noise” such as stego payloads and estimate the cover image statistics [27]. We use calibration to generate the POMM features for steganalysis, cropping by 4 pixels on each side. For an image $I^o$, we obtain a calibrated image $I^c$. The intra and inter block POMMs are calculated for both $I^o$ and $I^c$. The final feature vector $F$ is obtained by taking the difference between $F^0$ and $F^c$, $F = (F^o - F^c)(w, z)$. We get total of $2 \ast (2T + 1)^2$ features. For our experiments, we use threshold value $T = 3$. This results in 49 intrablock and 49 interblock features giving a total of 98 POMM features.
3.3.2 Histogram signature patterns (zero patterns)

As shown in section 1.3.2, when the primary quality factor is smaller than secondary quality factor, the mode histograms of DCT coefficients have zeros at some places. These zeros can be captured by looking at the forward differences between histogram bin values. For a single-compressed image, the distribution of DCT coefficients is ideally Gaussian. The forward differences are thus almost always small and positive. For double-compressed images, the zeros can take the form of local minimum instead of being exact zeros. So instead of finding absolute zeros, if the drop in the successive histogram values is more than 75%, it is considered to indicate presence of a zero. The value 75 was determined experimentally. For images with primary quality factor less than the secondary quality factor, this approach is followed. For secondary quality factor of 75, the primary quality factors under consideration are given by:

\[ s \in U_{75} = \{63, 68, 69, 70, 71, 72, 73\} \]

The quality factor 74 is not taken into account because the quantization steps for the 9 lower modes we use are identical to the quantization steps for quality factor 75. Thus images having
primary quality factor 74 and secondary quality factor 75 are considered as single-compressed for classification purposes.

When the secondary quality factor is 80, there are 13 primary quality factors under consideration which are given by:

\[ s \in U_{80} = \{63, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79\} \]

For any given image, we first inspect the first 22 bin values in the histograms of the nine low frequency DCT coefficients. The nine low frequency modes are:

\[ H = \{(0,1), (0,2), (0,3), (1,0), (1,1), (1,2), (2,0), (2,1), (3,1)\} \quad (3.3) \]

Remaining modes are not considered because most of the quantized DCT coefficients have value zero or close to zero. The histogram location \( k \) for mode \( m \) is represented by \( h_m(k) \) where \( 0 < (m = i+j) \leq 3 \); \( i \) and \( j \) are the coordinates of the location in each \( 8 \times 8 \) block.

Using the values from \( h_m(k) \), we create a 9 dimensional matrix \( \hat{M} \) for the candidate image.

\[ \hat{M}(m,k) = \chi_{\geq 0} \left( \frac{h_m(k) - h_m(k+1)}{h_m(k)} - 0.75 \right) \quad (3.4) \]

where \( 1 \leq m \leq 9 \) and \( 1 \leq k \leq 22 \).

Here, \( \chi \) is a thresholding function defined by

\[ \chi_{\geq T}(a) = \begin{cases} 1, & a \geq T \\ 0, & \text{otherwise} \end{cases} \quad (3.5) \]

Each value \( \hat{M}(m,k) \) is a zero feature. The matrix \( \hat{M} \) is called the zero feature matrix.

We use zero features also for:

1. Double-compression detection
2. Primary quality factor estimation.
3. Cover Vs. stego detection when the primary quality factor is smaller than the secondary quality factor.
3.3.2.1 Histogram zero features for double-compression detection

Double-compression detection is achieved using a combination of POMM features and zero features. We observed that the accuracy of the double-compression detector improves significantly when the zero features were added to the 98 POMM features. The zero features are particularly effective when the primary quality factor of an image is lower than the secondary quality factor.

For a given input image, a 9 x 21 sized matrix $\hat{M}$ of the zero locations is created using equation 3.4. Then the row values are summed to produce 9 features. These features are appended to the 98 POMM features and used to train the SVMs. Mathematically, a zero feature for the $i^{th}$ row of $\hat{M}$ is given by:

$$z_m = \sum_{k=1}^{21} \hat{M}(m,k)$$  \hspace{1cm} (3.6)

where $1 \leq m \leq 9$.

3.3.2.2 Primary Quality Factor Estimation

We observe that the signature patterns in the histograms are generally distinctive for a given combination of primary and secondary quality factors. These patterns of zeros were explained in detail in Section 1.3. We used this phenomenon for estimating the primary quality factor of the image. For a given combination of primary and secondary quality factors, we create standard matrices $M_t^s$, where $t$ is the secondary quality factor. These matrices capture the expected locations of zero-valued histogram bins. Thus, for primary quality factor $s$ and secondary quality factor $t, s < t$;

$$M_t^s(m,k) = \begin{cases} 
1, & \text{if bin } k \text{ of mode } m \text{ is zero} \\
0, & \text{if bin } k \text{ of mode } m \text{ is not zero}
\end{cases}$$ \hspace{1cm} (3.7)

For example, consider the case when primary quality factor is 63 and secondary quality factor is 75. The quantization matrices $Q^1$ and $Q^2$ are given below.
As described in section 1.2, the quantization steps at mode (2, 2) are 9 and 6 respectively. This results in the histogram zeros for bins 1, 4, 7, 10, 13 and so on. Similarly the zero patterns are obtained for other eight low frequency modes. A standard matrix $M_{63}$ is formed as follows:

$$Q^1 = \begin{bmatrix}
12 & 8 & 7 & 12 & 18 & 30 & 38 & 45 \\
9  & 9 & 10 & 14 & 19 & 43 & 44 & 41 \\
10 & 10 & 12 & 18 & 30 & 42 & 51 & 41 \\
10 & 13 & 16 & 21 & 38 & 42 & 51 & 41 \\
13 & 16 & 27 & 41 & 50 & 81 & 76 & 57 \\
18 & 26 & 41 & 47 & 60 & 77 & 84 & 68 \\
36 & 47 & 58 & 64 & 76 & 90 & 89 & 75 \\
53 & 68 & 70 & 73 & 83 & 74 & 76 & 73
\end{bmatrix} \quad Q^2 = \begin{bmatrix}
8 & 6 & 5 & 8 & 12 & 20 & 26 & 31 \\
6 & 6 & 7 & 10 & 13 & 29 & 30 & 28 \\
7 & 7 & 8 & 12 & 20 & 29 & 35 & 28 \\
7 & 9 & 11 & 15 & 26 & 44 & 40 & 31 \\
9 & 11 & 19 & 28 & 34 & 55 & 52 & 39 \\
12 & 18 & 28 & 32 & 41 & 52 & 57 & 46 \\
25 & 32 & 39 & 44 & 52 & 61 & 60 & 51 \\
36 & 46 & 48 & 49 & 56 & 50 & 52 & 50
\end{bmatrix}
$$

In a similar fashion, we generate a set of standard matrices $S_{std}$ for PQF $\in U_{75}$. When the secondary quality factor is 75, the standard matrices $M^s_{75}$ can be given as:

$$M^s_{63} = \begin{bmatrix}
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1
\end{bmatrix}$$

In a similar fashion, we generate a set of standard matrices $S_{std}^75$ for PQF $\in U_{75}$. When the secondary quality factor is 75, the standard matrices $M^s_{75}$ can be given as:

$$M^s_{75} = \{M_{63}, M_{68}, M_{69}, M_{70}, M_{71}, M_{72}, M_{73}\}$$

and when the secondary quality factor is 80, the set contains

$$M^s_{80} = \{M_{63}, M_{68}, M_{69}, M_{70}, M_{71}, M_{72}, M_{73}, M_{74}, M_{75}, M_{76}, M_{77}, M_{78}, M_{79}\}.$$
When a JPEG image is input to this algorithm, data matrix $\hat{M}$ is derived as shown in equation 3.4. $\hat{M}$ is used for pattern matching to determine which standard matrix from the set $S_{std}$ of standard matrices matches most closely with it. The closest estimated standard matrix $M_{est}^t$ is formed by minimizing the absolute difference between each $M_s^t$ and $\hat{M}$:

$$M_{est}^t = \arg \min_{M_s^t \in S_{std}} \sum |M_s^t - \hat{M}|$$  \hspace{1cm} (3.8)

It is also observed that the dips in the histograms assume absolute zero value whereas for stego images, the dips can be characterized by drop in the histogram values. This fact can be used in the successful detection of stego images.

### 3.3.2.3 Cover Vs. stego detection for PQF < SQF

The histogram zero patterns can also be used for stego detection in images with PQF < SQF. It was stated in section 1.3 that the 75% drop in the histogram bin values characterizes a zero for a double-compressed image. Ideally a zero bin should have value exactly equal to zero but the statistics of the DCT coefficients change due to the process of embedding. In order to account for these changes, 75% drop is considered to indicate a zero. For double-compressed cover images, the drop is 100% and the zero bins actually take zero value. For double-compressed stego images, we consider even a small drop in the bin value as a zero. This acts as a distinguishing feature between cover and stego images. For all the standard matrices $M_s$ from the standard set $S_{std}$, we count the number of absolute zero bins that are expected to occur for each combination of PQF and SQF.

For an input image, first the PQF is estimated as explained in section 3.3.2.2. For the estimated PQF, the expected count of zero bins is obtained. From matrix $\hat{M}$ derived from an input image, we count the total number of absolute zero values. If this count matches exactly with the expected standard count, the image is a cover image; otherwise it is a stego image.

In this case, classification is done using an analytical method as opposed to SVM-based detection. For a given detector, set $S_{std}$ of standard matrices is created and stored beforehand. This method saves a lot of time required in feature extraction and SVM training.
3.4 Comparison with the state-of-the-art double-compression detectors

Next, we describe two state-of-the-art detectors and compare their performance with our detector.

3.4.1 Double-compression Detector proposed by T. Pevny et al.

The first state-of-the-art detector considered for comparison is proposed by T. Pevny et al. in [21]. The features are derived from mode histograms of the quantized DCT coefficients in the JPEG image. Nine low frequency modes are considered for constructing the feature vector. These modes are the coefficients located at the nine low frequency locations in the 8 arrays as given by $H$ in equation 3.3.

First sixteen multiples of each secondary quantization step $Q_{ij}^2$ are extracted which results in a feature vector given below:

$$\overrightarrow{x} = \frac{1}{C_{ij}}(h_{ij}(0), h_{ij}(1), ..., h_{ij}(15)) \quad (3.9)$$

where $0 < (i + j) \leq 3$, $h_{ij}(m)$ is the frequency of occurrence of $(m.Q_{ij}^2)$ in the histogram of mode $(i, j)$, and $C_{ij}$ is the normalizing constant

$$C_{ij} = \sum_{m=0}^{15} \lim_{m=0} h_{ij}(m).$$

These feature vectors are used to train the double-compression detector which after training, classifies an input image as one of the two classes: double compressed, or not. The Support Vector Machine (SVM) was used as a pattern classifier [9]. The SVM is a soft-margin support vector machine (C-SVM) that uses Gaussian kernel. It classifies the images into the two classes, single-compressed or double-compressed. Primary quality factors from the set $S$ are used for training and the secondary quality factor is fixed at 75 since it is the default quality factor for outguess.

3.4.2 Double-compression Detector proposed by Chen et al.

Another state-of-the-art double-compression detector used for comparison is due to Chen et al. [22]. In this method, nearest-neighbour differences in the vertical, horizontal, major diagonal
and minor diagonal directions are calculated, resulting in a set of four 2-dimensional difference arrays. In order to reduce the computations, these arrays are thresholded to values in between -4 and +4. The difference arrays are modeled using one-step Markov process characterized by a transition probability matrix (TPM). This results in 4 9 × 9 TPM matrices which leads to 324 points feature vector. These features are used to train a support vector machine. See [22] for more details.

The authors assume that the double-compressed images are cover images only and the primary quality factors considered are in the range 50 to 95 in steps of 5. The detection accuracies obtained are very high. However, when the method is used for double-compressed stego images, the detection accuracies drop. The results of double-compression detection are presented in chapter 4 and compared with the Pevny’s detector and our POMM based detector.
CHAPTER 4. EXPERIMENTS AND RESULTS

In this chapter, we describe the experimental set up and present the results of classification. We have constructed a complete steganalyzer system that consists of a double-compression detector, a PQF splitter, two cover Vs. stego detectors and two PQF detectors. The first two modules are based on SVM training using the POMMs and histogram zeros features. For all the other modules, we use analytical methods which do not require time consuming feature extraction and SVM training steps. These novel methods are a major contribution of this research.

4.1 Double-compression Detector (DCDetector)

An unknown image is input to the steganalyzer. If the observed quality factor of the image is 75 or 80, it is likely to be double-compressed. This image is passed on to the double-compression detector (referred to as DCDetector). Two separate detectors are created for secondary quality factors 75 and 80. Each detector is a binary classifier trained on 15000 single-compressed and 15000 double-compressed image data. For the first detector, the images with PQF 74 and 75 are excluded whereas for the second detector, images with PQF 80 are excluded from the training database. This is because statistics of the images with PQF equal to SQF is the same as single-compressed images. Figure 4.1 shows the database used for training the SVM for SQF 75 case. There are 15000 images of each class. The stego images are further divided amongst the number of classes and subdivided amongst three embedding levels. Double-compressed images are also divided equally amongst the different primary quality factors as shown.
Figure 4.1  Database used for training double-compression detector

For testing, we use 1500 cover images at each PQF in the range 63 to 94. In case of f5 and outguess, we use 500 images at each of the three embedding levels, resulting in 1500 images at each PQF.

Our double-compression detector uses combination of 98 POMM features and 9 zero features giving a total of 107 features. A double-compression detector is required to have very low false positive rate. False positive rate is the percentage of single-compressed test images which get classified as double-compressed. If a single-compressed image is classified as double-compressed, it can be assigned to only cover, f5 or outguess instead of six classes: cover, f5, jp hide & seek, jsteg, outguess and steghide. This leads to classification errors.

Figure 4.2 shows the accuracy plots for the detector when tested on double-compressed cover, f5 and outguess images. Plots on the left represent results for SQF 75 whereas lplots on the right represent results for SQF 80.

Table 4.1 shows the false positive rates obtained by testing the DC Detector on single compressed images.

Discussion of the results:

1. In Figure 4.2, it can be seen that the DCDetector can classify cover images accurately except for two cases. First case is when the PQF is equal to or close to the SQF (74, 75 and 76 in the left column plots and 80 and 81 in the right column plots) and second case is when PQF is greater than 90. PQF = 74 and SQF = 75 is a special case because the nine low frequency locations that we consider are identical for quality factors 74 and
Figure 4.2  Accuracy of double-compression detectors on double-compressed cover [(a),(b)], f5 [(c),(d)] and outguess [(e),(f)] images
75. This makes it impossible to detect images with this combination of quality factors using only this information. This is because when the PQF is very close to the SQF, the statistics of double-compressed image is close to the corresponding single-compressed image. Also, for PQF greater than 90, most of the quantization steps are equal to one due to which the effect of double-compression is not obvious. This justifies the drop in the detection accuracies in these two cases.

2. Similar trend is observed in case of f5 and outguess with respect to the PQF. Also, the detection accuracies drop when the embedding level increases.

3. For the cases mentioned above where the PQF is close to the SQF, if the detector classifies the image as single-compressed, it is considered as a correct decision. In order to determine the overall detection accuracy, we consider the average of true positive rate for double-compressed test images and true negative rate for the single-compressed test images.

4. Table 4.1 shows the false positive rates. Except for f5 at 0.4 bpnz and outguess at 0.2 bpnz, the false positive rates are lower than 8% for SQF = 75. For SQF = 80, the false positive rates are lower than corresponding cases of SQF = 75 by at least \( \frac{1}{2} \).

5. Overall performance of detector for SQF = 80 is better than the detector for SQF = 75. This is given in Table 4.2. The overall accuracies are calculated by averaging the accuracies over all PQFs and all embedding levels.

<table>
<thead>
<tr>
<th>(a) SQF = 75</th>
<th>% False Positive Rate</th>
<th>(b) SQF = 80</th>
<th>% False Positive Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cover</td>
<td>3</td>
<td>1.8</td>
<td></td>
</tr>
<tr>
<td>F5</td>
<td>2.8</td>
<td>1</td>
<td>2.8</td>
</tr>
<tr>
<td>JPHS</td>
<td>3.4</td>
<td>0.4</td>
<td>0.8</td>
</tr>
<tr>
<td>Jsteg</td>
<td>1.2</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>Ouguess</td>
<td>3</td>
<td>0.2</td>
<td>0.6</td>
</tr>
<tr>
<td>Steghide</td>
<td>3</td>
<td>0.4</td>
<td>0.4</td>
</tr>
</tbody>
</table>

Table 4.1 False positive rate of double-compression detector for (a) SQF = 75 and (b) SQF = 80.
Table 4.2 Average accuracy of DC Detectors for double-compressed images

<table>
<thead>
<tr>
<th></th>
<th>SQF = 75</th>
<th>SQF = 80</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cover</td>
<td>94.65</td>
<td>97.08</td>
</tr>
<tr>
<td>F5</td>
<td>88.86</td>
<td>90.23</td>
</tr>
<tr>
<td>Outguess</td>
<td>92.22</td>
<td>91.08</td>
</tr>
</tbody>
</table>

Comparison with the state-of-the-art double-compression detectors:

We coded up the state-of-the-art double-compression detectors by Pevny [6] and Chen [22] and compared their performance with our POMM based detector. We implemented the algorithms given in their papers to obtain the detection accuracies using the same database as for the POMMs. Figure 4.3, Figure 4.4 and Figure 4.5 show the comparison of detection accuracies of these detectors on the double-compressed cover, f5 and outguess images respectively. The left column shows the comparisons for SQF = 75 case whereas the right column shows the comparisons for SQF = 80 case.

Table 4.3 contains the comparison of false positive rates of the three detectors.

Discussion of results:

1. Both the state-of-the-art detectors follow the same trends with respect to PQF as POMM-based detector. The detection accuracies drop when PQF is equal to or close to SQF.
Figure 4.4 Comparison of accuracy of double-compression detectors for double-compressed f5 images for embedding levels 0.05 bpnz [(a),(b)], 0.1 bpnz [(c),(d)] and 0.4 bpnz [(e),(f)]
Figure 4.5  Comparison of accuracy of double-compression detectors for double-compressed outguess images for embedding levels 0.05 bpnz [(a), (b)], 0.1 bpnz [(c), (d)] and 0.2 bpnz [(e), (f)]
The accuracies also drop for PQF greater than 90 due to the reason explained previously.

2. Amongst the three detectors under consideration, Pevny’s detector has the lowest accuracies.

3. Chen’s detector has accuracies comparable to our POMM-based detector for a limited number of PQFs. For most of the PQFs, the accuracies are lower than our detector.

4. Our POMM detector has the lowest false positive rate (FPR) when SQF = 80. For most of the cases, the FPR is less than 1%.

5. Table 4.4 shows the average detection accuracies for the three detectors under consideration. These numbers are the accuracies averaged over all the PQFs and all the embedding levels. We can see that the average detection accuracies of our detector are better than the state-of-the-art detectors.

### 4.2 Primary Quality Factor Splitter (PQFSplitter)

If the DC Detector classifies the image as double-compressed, we use two different approaches depending on the Primary Quality Factor (PQF) and the Secondary Quality Factor.
Table 4.4 Average detection accuracies of Pevny, Chen and POMM DC Detectors for (a) SQF = 75 and (b) SQF = 80

<table>
<thead>
<tr>
<th></th>
<th>SQF = 75</th>
<th>Average Accuracy %</th>
<th>SQF = 80</th>
<th>Average Accuracy %</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pevny</td>
<td>Chen</td>
<td>POMM</td>
<td>Pevny</td>
</tr>
<tr>
<td>Cover</td>
<td>27.37</td>
<td>63.82</td>
<td>94.85</td>
<td>22.38</td>
</tr>
<tr>
<td>F5</td>
<td>34.62</td>
<td>52.16</td>
<td>88.86</td>
<td>29.44</td>
</tr>
<tr>
<td>Outguess</td>
<td>31.5</td>
<td>51.9</td>
<td>92.22</td>
<td>26.67</td>
</tr>
</tbody>
</table>

Figure 4.6 Database used for training PQF splitter

(SQF) of the image. This works because the signature patterns in the histograms of the quantized DCT coefficients are different when the PQF is smaller than the SQF and when the PQF is greater than the SQF. In order to decide which approach to take, it is necessary to find out the relation between PQF and SQF. This is done by another binary classifier called PQF Splitter. The PQF Splitter classifies the images based on whether the PQF is smaller than or greater than the SQF. Two separate binary classifiers are created for SQF of 75 and 80. For SQF = 75, there are 7 PQFs in the range below 75 (63, 68, .., 73) and 19 PQFs above 75 (76, 77,.., 94). We use 15000 training images for each class. The images are further divided into cover, f5 and outguess categories and then subdivided based on the PQF and embedding levels as well. This division for database creation is depicted in Figure 4.6. In the same way, a database is created for SQF = 80 case. In this case, there are 13 PQFs below 80 and 14 PQFs above 80. So the division of images changes accordingly.
The classification accuracies of the PQF Splitters are shown in Figure 4.7. The plots on the left show accuracies for SQF = 75 and the plots on the right show accuracies for SQF = 80 case.

**Discussion of the results:**

1. When SQF = 75, Figure 4.7 shows the detection accuracies are almost always close to 100% over the entire range of PQFs for cover, f5 and outguess.

2. When SQF = 80, the detection accuracies for cover images are close to 100% over the entire range of PQFs.

3. When SQF = 80, the detection accuracies for f5 and outguess images drop when PQF is 79. In this case, the double-compressed image is close to its single-compressed version.

4. For PQFSplitter, the detection accuracies do not vary with the embedding levels.

### 4.3 Cover Vs. stego detector

Once the range of PQF is determined by the PQF splitter, we perform cover Vs. stego detection. Depending on whether the PQF of an image is smaller or larger than SQF, we use different methods for classification. Sections 4.3.1 and 4.3.2 contain detailed discussions about the two methods.

#### 4.3.1 Cover Vs. stego detector for Primary Quality Factor $<$ Secondary Quality Factor

When PQF of an image is less than SQF, an analytical method described in section 3.3.2 is used. The method is based on the histogram signature patterns. This approach saves time and computational resources required for the feature extraction process and intensive training of support vector machines. Figure 4.8 shows the detection accuracy for when the observed quality factor (SQF) is 75 (left column) and for when the observed quality factor is 80 (right column).
Figure 4.7  Accuracy of PQF Splitters on double-compressed Cover, F5 and Outguess images
Figure 4.8 Accuracy of cover Vs stego detector for PQF < SQF
Discussion of the results:

1. We observe that the detection accuracies are almost always close to 100% for f5 and outguess.

2. For cover, the detection accuracies are close to 100%. When the PQF is very close to SQF, there is a drop in the detection accuracy.

4.3.2 Cover Vs. stego classifier for Primary Quality Factor > Secondary Quality Factor

If the PQF is greater than SQF, we propose and compare following three methods.

4.3.2.1 Cover Vs. stego binary classifier

A support vector machine is trained for binary classification. In this case, f5 and outguess images together form the ‘stego’ class whereas ‘cover’ images form another class. In this case, we do not determine the steganographic algorithm used for embedding.

4.3.2.2 Multi-classifier

In this method the support vector machine is used as a multi-classifier and the input images are assigned to one of the three classes: cover, f5 or outguess.

4.3.2.3 Three Binary Classifiers combined using majority vote

In this method, three binary classifiers are created:

- Classifier1: Cover Vs F5
- Classifier2: Cover Vs Outguess
- Classifier3: F5 Vs Outguess

The final result of classification is obtained by majority voting.

These three methods are tested for the SQF = 75 case. Figure 4.9 shows the comparison of the detection accuracies in three cases.
Figure 4.9 Comparison of three detectors for cover Vs. stego detection for PQF > SQF, (SQF = 75)
It can be seen that the detection accuracy is the highest when the majority vote method is used. This approach is therefore selected and similar classifiers are created for SQF = 80 case. In Figure 4.10, we show the accuracies of the cover Vs. stego detectors when PQF is greater than SQF. The plots on the left represent SQF = 75 case whereas the plots on the right represent SQF = 80 case.

We summarize the results of cover Vs. stego detection for PQF > SQF case in the confusion matrix given in Table 4.5. We pass a test images of known class to the detector and calculate the probabilities of both true and false detection. We observe that the detection accuracies for cover and outguess are above 96%. For f5, 9.5% of the images get misclassified as cover when SQF is 75 and this corresponds to the lowest detection accuracy. The low accuracies for f5 are due to the fact that f5 does not preserve the global statistics of the image. The artifacts of double-compression are thus lost during the process of embedding.

At this point, we have cover Vs. stego detectors for both PQF < SQF and PQF > SQF case. We determine the overall accuracy of the detector by averaging the accuracies of these two detectors over all PQFs and all embedding levels. This gives us the cover Vs. stego detection accuracy of our detector for double-compressed images. In Table 4.6, we compare these results with those obtained from previous Canvass software, given in Table 1.1. We can clearly see significant rise in the detection accuracies.
Figure 4.10  Accuracy of cover Vs stego detector for PQF > SQF
Table 4.6  Overall detection accuracy of (a) previous Canvass (b) our detector tested on double-compressed images

<table>
<thead>
<tr>
<th>SQF = 75</th>
<th>0.05 bpnz</th>
<th>0.1 bpnz</th>
<th>0.4 bpnz$^*$</th>
<th>Average</th>
<th>FPR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cover</td>
<td>45.39</td>
<td>45.39</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F5</td>
<td>28.84</td>
<td>34.54</td>
<td>58.29</td>
<td>40.56</td>
<td></td>
</tr>
<tr>
<td>Outguess</td>
<td>42.04</td>
<td>57.73</td>
<td>74.84</td>
<td>58.02</td>
<td></td>
</tr>
</tbody>
</table>

(b)

<table>
<thead>
<tr>
<th>SQF = 75</th>
<th>% Detection Accuracy of new detectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cover</td>
<td>97.45</td>
</tr>
<tr>
<td>F5</td>
<td>82.28, 90.88, 95.36</td>
</tr>
<tr>
<td>Outguess</td>
<td>93.91, 96.29, 90.77</td>
</tr>
</tbody>
</table>

4.4 Primary Quality Factor Detector

The last step in the blind steganalysis is estimation of the primary quality factor. It can be used to extract further information regarding the secret payload; although we do not use it for further processing in this work. The signature zero patterns in the histograms are used for the PQF detection. Separate detectors are created for SQF = 75 and SQF = 80 case.

Figure 4.11 shows the detection accuracy plots. Again, the plots on the left represent SQF = 75 case and plots on the right represent SQF = 80 case.

Discussion of results:

We first discuss the results when SQF = 75. These are shown in the left column of 4.11.

1. The PQF detection accuracies for double-compressed cover images are almost always 100% when PQF < SQF. In general, when PQF is less than the SQF, the zero patterns in the histograms are prominent. This leads to high detection accuracies.

2. Double-compressed f5 images with PQF 70 get misclassified as PQF = 71. Out of the nine low-frequency modes considered for the detection, the quantization matrices for quality factors 70 and 71 vary for 2 modes.
Figure 4.11 Accuracy of PQF detector
3. We do not detect PQF 89. This is because the histogram patterns arising from the combination of PQFs 88 and 89 with SQF 75 are identical. The quality factor 89 gets detected as 88. Therefore we exclude it from the detection algorithm.

4. In general, the detection accuracies are low when PQF > SQF because the histogram patterns are not very prominent.

We now discuss the results for SQF = 80 case. These are shown in the right column of 4.11.

1. Similar to the SQF = 75 case, the detection accuracies are close to 100% when PQF < SQF.

2. PQF 75 gets detected as 74. This is because the values at nine-low frequency modes under consideration are identical for these two quality factors. Therefore we exclude the quality factor 75. Similarly, for f5 and outguess, PQF 85 almost always gets detected as 84 and PQF 88 gets detected as 89. This explains the drops in the detection accuracies for these PQFs.

3. We do not detect PQFs 92 and 93. This is because the histogram patterns arising from the combinations of PQFs 91, 92 and 93 with SQF 80 are identical. The quality factors 92 and 93 get detected as 91. Therefore we exclude these quality factors from the detection algorithm.
CHAPTER 5. CONCLUSIONS AND FUTURE GOALS

In this thesis, we created a complete steganalyzer system for single as well as double- compressed images. We introduced a new statistical modeling tool to measure the changes caused by various steganographic algorithms as well as by double-compression. We showed that the POMM features perform better than the state-of-the-art double-compression detectors. We also proved the utility of POMMs for solving variety of classification problems such as double-compression detection, PQF splitting and cover Vs. stego detection.

We introduced analytical methods for cover Vs. stego detection and primary quality factor detection. The methods are based on the signature patterns in the histograms of quantized DCT coefficients, as opposed to the other SVM-based classification methods. Each SVM in our experiments is trained on 30000 image data and tested on approximately 1,35,000 image data for different cases. Creating each SVM involves 1 day to extract the training features from 30000 images, 1.5 days of actual SVM training and 2 days for testing. The SVMs-based approach [6] requires 1 SVM for double-compression detector, 159 SVMs for primary quality step estimation, 3 binary classifiers for stego detection of double-compressed images and 15 SVMs for stego detection of single-compressed images. Thus a total of 178 SVMs are required. Our approach on the other hand uses 1 SVM for double-compression detection, 1 for PQF splitting, 15 SVMs for single-compressed stego detection and 3 SVMs for double-compressed stego detection, which gives a total of 20 SVMs. These novel analytical methods thus save a large amount of time required for feature extraction and intensive SVM training. The histogram pattern features were also used in addition to POMMs to improve the detection accuracies of SVM-based classifiers.

We show that the detection scheme works better if a double-compression detector is used as a pre-classifier. The detection accuracies for double-compressed images improve significantly
compared to the previous Canvass. We also compare the performance of the various modules with those presented in [6] and [21]. The POMM-based PQF detector has high detection accuracies for PQF < SQF.

The conditional probabilities given by POMMs describe the relations between inter and intrablock pixels in a JPEG image. In the future, other functions could be investigated to describe other pixel dependencies. Also, currently we limit the steganalysis of single-compressed images to quality factor 75 and that of double-compressed images to secondary quality factors of 75 and 80. But this novel method for steganalysis of double-compressed data looks promising and could be generalized for any combination of primary and secondary quality factors.
BIBLIOGRAPHY


