Fall 2017

Machine learning based digital image forensics and steganalysis

Guanshuo Xu
New Jersey Institute of Technology

Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

MACHINE LEARNING BASED DIGITAL IMAGE FORENSICS AND STEGANALYSIS

by
Guanshuo Xu

The security and trustworthiness of digital images have become crucial issues due to the simplicity of malicious processing. Therefore, the research on image steganalysis (determining if a given image has secret information hidden inside) and image forensics (determining the origin and authenticity of a given image and revealing the processing history the image has gone through) has become crucial to the digital society.

In this dissertation, the steganalysis and forensics of digital images are treated as pattern classification problems so as to make advanced machine learning (ML) methods applicable. Three topics are covered: (1) architectural design of convolutional neural networks (CNNs) for steganalysis, (2) statistical feature extraction for camera model classification, and (3) real-world tampering detection and localization.

For covert communications, steganography is used to embed secret messages into images by altering pixel values slightly. Since advanced steganography alters the pixel values in the image regions that are hard to be detected, the traditional ML-based steganalytic methods heavily relied on sophisticated manual feature design have been pushed to the limit. To overcome this difficulty, in-depth studies are conducted and reported in this dissertation so as to move the success achieved by the CNNs in computer vision to steganalysis. The outcomes achieved and reported in this dissertation are: (1) a proposed CNN architecture incorporating the domain knowledge of steganography and

steganalysis, and (2) ensemble methods of the CNNs for steganalysis. The proposed CNN is currently one of the best classifiers against steganography.

Camera model classification from images aims at assigning a given image to its source capturing camera model based on the statistics of image pixel values. For this, two types of statistical features are designed to capture the traces left by in-camera image processing algorithms. The first is Markov transition probabilities modeling block-DCT coefficients for JPEG images; the second is based on histograms of local binary patterns obtained in both the spatial and wavelet domains. The designed features serve as the input to train support vector machines, which have the best classification performance at the time the features are proposed.

The last part of this dissertation documents the solutions delivered by the author's team to The First Image Forensics Challenge organized by the Information Forensics and Security Technical Committee of the IEEE Signal Processing Society. In the competition, all the fake images involved were doctored by popular image-editing software to simulate the real-world scenario of tampering detection (determine if a given image has been tampered or not) and localization (determine which pixels have been tampered). In Phase-1 of the Challenge, advanced steganalysis features were successfully migrated to tampering detection. In Phase-2 of the Challenge, an efficient copy-move detector equipped with PatchMatch as a fast approximate nearest neighbor searching method were developed to identify duplicated regions within images. With these tools, the author's team won the runner-up prizes in both the two phases of the Challenge.

MACHINE LEARNING BASED DIGITAL IMAGE FORENSICS AND STEGANALYSIS

by
Guanshuo Xu

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Electrical Engineering

Helen and John C. Hartmann Department of Electrical and Computer Engineering

January 2017

.

APPROVAL PAGE

MACHINE LEARNING BASED DIGITAL IMAGE FORENSICS AND STEGANALYSIS

Guanshuo Xu

Dr. Yun Q. Shi, Dissertation Advisor — Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Atam P. Dhawan, Committee Member — Date
Vice Provost R&D, NJIT
Distinguished Professor of Electrical and Computer Engineering, NJIT

Dr. Richard A. Haddad, Committee Member — Date
Professor Emeritus of Electrical and Computer Engineering, NJIT

Dr. Mengchu Zhou, Committee Member — Date
Distinguished Professor of Electrical and Computer Engineering, NJIT

Dr. Frank Y. Shih, Committee Member — Date
Professor of Computer Science, NJIT

BIOGRAPHICAL SKETCH

Author: Guanshuo Xu

Degree: Doctor of Philosophy

Date: January 2017

Undergraduate and Graduate Education:

- Doctor of Philosophy in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2017

- Master of Science in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2008

- Bachelor of Science in Electrical Engineering,
  Tongji University, Shanghai, P. R. China, 2006

Major: Electrical Engineering

Presentations and Publications:

J. Ye, G. Xu, and Y. Q. Shi, "A Convolutional Neural Network Based Deep Learning Architecture for Seam Carving Detection," *IEEE Signal Process. Lett.*, Dec. 2016, submitted.

G. Xu, H. Wu, and Y. Q. Shi, "Ensemble of CNNs for steganalysis: An empirical study," in *Proc. 4th ACM Workshop Inf. Hiding Multimedia Secur. (IH&MMSec)*, 2016, pp. 103–107.

G. Xu, H. Wu, and Y. Q. Shi, "Structural design of convolutional neural networks for steganalysis," *IEEE Signal Process. Lett.*, vol. 23, no. 5, pp. 708–712, Mar. 2016.

G. Xu, J. Ye, and Y. Q. Shi, "New developments in image tampering detection," in *Proc. 13th Int. Workshop Forensics Watermark. (IWDW)*, 2014, pp. 3–17.

G. Xu and Y. Q. Shi, "Camera model identification using local binary patterns." in *Proc. IEEE Int. Conf. Multimedia and Expo. (ICME)*, 2012, pp. 392–397.

S. Gao, G. Xu, and R. Hu, “Camera model identification based on the characteristic of CFA and interpolation,” in *Proc. 10th Int. Workshop Forensics Watermark. (IWDW*), 2011, pp. 268–280.

G. Xu, Y. Q. Shi, and W. Su, “Camera brand and model identification using moments of 1-D and 2-D characteristic functions,” in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, 2009, pp. 2917–2920.

G. Xu, S. Gao, and Y. Q. Shi, “Camera-model identification using Markovian transition probability matrix,” in *Proc. 8th Int. Workshop Forensics Watermark. (IWDW*), 2009, pp. 294–307.

< 献给我的家人 >
< To my beloved family >

ACKNOWLEDGMENT

I would like to express my gratitude to my advisor, Professor Yun Q. Shi, for his tremendous assistance with this dissertation. From the beginning to the end, he has been a steadfast source of information, ideas, support, and energy. I am deeply grateful for his guidance, patience, and encouragement and I will be forever grateful for his trust and support that this dissertation could be completed.

I would also like to express my appreciation to the distinguished members of the dissertation committee: Drs. Richard A. Haddad, Atam P. Dhawan, Mengchu Zhou, and Frank Y. Shih, for their active participation and valuable comments.

Many of my present and former colleagues in the Electrical and Computer Engineering Department at NJIT and visiting scholars from outside of NJIT are deserving recognition for their help during my graduate study life.

The first part of the research reported in the dissertation has been partly sponsored by the NJIT Faculty Seed Grant in the fall of 2014.

Finally, I would like to thank my parents for their support and encouragement. All have been encouraging; I would quite simply not have completed this dissertation without their support and encouragement.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

LIST OF TABLES
(Continued)

LIST OF FIGURES

LIST OF FIGURES
(Continued)

LIST OF FIGURES
(Continued)

Figure **Page**

LIST OF FIGURES
(Continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Overview

Digital images have become one of the major information carriers in our modern daily lives. While people enjoy the efficiency of information exchange, the security and trustworthy of digital images have become a crucial issue due to the ease of malicious processing, e.g., embedding secret messages for covert communications, altering origin and content of images with popular image editing software. These malicious usages could give rise to serious problems if they are taken advantage of by terrorist organizations, treated as evidence in court, or published by mass media for information dissemination. Therefore, the study and research on image steganalysis — determining if a given image contains secret information, and on image forensics — determining the origin and authenticity of a given image as well as revealing the processing history it has gone through, have become crucial to our digital society.

In this dissertation, steganalysis and forensics of digital images are mainly treated as classification problems so as to make advanced machine learning (ML) methods applicable. Three topics are covered: (1) architectural design of convolutional neural networks (CNNs) for steganalysis; (2) design of statistical features for camera model classification; and (3) real-world tampering detection and localization.

1.2 Contributions Made in This Dissertation Research

ML-based steganlaysis aims to distinguish images with secret messages embedded in, and their corresponding cover images. Since advanced steganography alters the pixel values in the image regions that are hard to detect during embedding, traditional ML-based steganalysis heavily relied on sophisticated manual feature design has been pushed to the limit. To overcome this difficulty, in-depth studies have been conducted by the author to move the success achieved by the CNNs from computer vision to steganalysis. The outcomes are (1) a proposed CNN architecture incorporating the domain knowledge of steganography and steganalysis, and (2) ensemble methods of the designed CNNs for steganalysis. The proposed CNN is currently the best classifier against advanced steganography; and its size is easily expendable for even better performance when better hardware is available. Unlike traditional feature-based methods that have been limited by the difficulty of manual feature design, the CNN-based classifiers jointly optimize feature extraction and classification; hence, they are expected to be future trend of multimedia forensics and steganalysis.

Camera model classification, aiming at assigning a given image to its source capturing device based on the statistics of pixel values, belongs to source identification in image forensics. For this, two types of statistical features have been proposed to capture the traces left by in-camera image processing algorithms of different makes and models. The first type is Markov transition probabilities of the neighboring block-DCT coefficients for JPEG images, the second is based on histograms of local binary patterns (LBPs) obtained in both the spatial and wavelet domains of images. The designed feature sets serve as the input to support vector machines for classification. These works were

done in the early stage of the Ph.D. research. While they have been surpassed by more recent methods, both of the two features sets achieved top performance at the time they were proposed. In the future, all of those feature-based methods are expected to be replaced by CNN-based methods.

The last part of this dissertation documents the solutions delivered by the author's team to The First Image Forensics Challenge organized by the Information Forensics and Security Technical Committee of the IEEE Signal Processing Society. In contrast to the common image tampering detection dataset created in a fully-controlled manner for pure research purposes, all the fake images involved in the challenge had been doctored by popular image-editing software to simulate the real-world scenario of tampering detection (images have been tampered or not) and localization (which pixels have been tampered); hence, the detection algorithms are required to be practical. For image-level tampering detection (Phase-1 of the challenge), we migrated advanced steganalysis features for tampering detection, which again prove that feature-based tampering detection methods work well in practice and features designed for steganalysis are applicable for tampering detection. For pixel-level tampering localization (Phase-2 of the challenge), having aware of the limitations of existing copy-move detection methods, we developed an efficient copy-move detector that employs PatchMatch as a fast approximate nearest neighbor searching method to identify duplicated regions for tampering localization. With these tools, the author's team won the runner-up prizes in both the two phases of the Challenge. Results show that there is still a lot of room for improvement, particularly for the localization problem.

1.3 Outline of This Dissertation

In Chapter 2, the motivation of using convolutional neural networks (CNNs), the architectural design of CNNs, and the ensemble study of CNNs for steganalysis are described in detail. Chapter 3 elaborates the Markov-based and the LBP-based feature sets designed for camera model classification. Chapter 4 documents our 2nd-place solutions for The First Image Forensics Challenge hosted by IEEE Signal Processing Society. Chapter 5 summarizes this dissertation.

CHAPTER 2

CONVOLUTIONAL NEURAL NETWORKS FOR STEGANALYSIS

2.1 Steganography and Steganalysis

Modern steganography can be used to embed secret messages into digital media for covert communications. Unlike cryptography, steganography hides the existence of the secret messages to the public except the intended recipients. This feature, together with the popularity of digital media as suitable covers and the huge amount of publicly available steganographic software, facilitate steganography for possible illegal and malicious usages. Real-life examples include al Qaeda's plan for attacks hidden in a pornographic video[1], covert communications inside a Russian spy ring in US with messages hidden in images using customized steganography software[2], the distribution of child pornography[3] and malicious software[4], etc. All the aforementioned examples have pointed to the urgent need of the counterpart of steganography – steganalysis.

As digital images are unarguably one of the most popular forms of multimedia on cyberspace, in this chapter, we focus our research on the advancement of steganalysis to fight steganography with digital still images. More precisely, we concentrate on detecting advanced steganography embedding in the original spatial domain of grayscale images. It has been demonstrated that the success of steganalysis on grayscale images in spatial domain could be extended to the steganalysis on JPEG format [86, 103] and color images [104, 105].

[1] http://www.cnn.com/2012/04/30/world/al-qaeda-documents-future (accessed on November 30, 2016)
[2] http://www.justice.gov/sites/default/files/opa/legacy/2010/06/28/062810complaint2.pdf (accessed on November 30, 2016)
[3] http://www.antichildporn.org/steganog.html (accessed on November 30, 2016)
[4] http://www.voanews.com/content/hackers-hiding-malware-in-plain-sight/2913694.html (accessed on November 30, 2016)

Both steganography and steganalysis have been studied for years, to better focus, only aspects of steganography and steganalysis closely related to our research will be covered in this dissertation. In Section 2.1.1, the information-theoretic framework of content-adaptive steganography will be presented. Feature-based steganalysis will be introduced in Section 2.1.2.

2.1.1 Advanced Steganography

While steganalysis is the focus in our works, having some knowledge of the data embedding methods would benefit the research on steganalysis.

Steganography aims to maximize the amount of embedded data hidden into images while minimizing the chance of being detected by either visual attack or statistical attack (steganalysis). As the changes of pixel values during embedding usually happen at the lowest bit-planes of pixel values to ensure visual imperceptibility, it is assumed that the detectability depends only on steganalysis, which relies on statistics of pixel values.

Steganography has been formalized as a rate-distortion problem [1, 2]. Given a *n*-pixel cover image $\mathbf{x} = (x_i)_{i=1}^n \in \mathcal{I}^n$, where $x_i \in \mathcal{I} = \{0,...,255\}$ for 8-bit grayscale images[5], the corresponding stego (message embedded) image $\mathbf{y} = (y_i)_{i=1}^n \in \mathcal{Y}$ is treated as a random variable $\mathbf{Y} \sim p(\mathbf{y}) \triangleq \Pr(\mathbf{Y} = \mathbf{y} \mid \mathbf{x})$, where $\mathcal{Y} = \mathcal{I}_1 \times \mathcal{I}_2 \times ... \times \mathcal{I}_n$ is the set of all possible *n*-pixel image **y** that **x** can be transformed to. In this dissertation, only ternary embedding[6] (±1 embedding) is considered, i.e., $\mathcal{I}_i = \{x_i - 1, x_i, x_i + 1\}$.

As is customary in research, the size of embedded message (payload) is fixed beforehand, and the most secure (the least detectable) steganography is desired with the

[5] For simplicity of notation, one-dimensional representation is used here for images.
[6] The saturation conditions when $x_i = 0$ or 255 are not considered here.

fixed payload. Previous research works have demonstrated that a promising approach to quantize detectability is to design a distortion function measuring how much the stego image differs from the original cover. Given that the distortion of the transformation from **x** to **y** is $D(\mathbf{y}) \triangleq D(\mathbf{x},\mathbf{y})$, embedding m bits into **x** on average while minimizing the average distortion can be formalized as

$$\begin{aligned} &\underset{p}{\text{minimize}}\, E_p\left[D\right] = \sum_{\mathbf{y}\in\mathcal{Y}} p(\mathbf{y})D(\mathbf{y}) \\ &\text{subject to } H(p) = -\sum_{\mathbf{y}\in\mathcal{Y}} p(\mathbf{y})\log_2 p(\mathbf{y}) = m\,. \end{aligned} \tag{2.1}$$

Note that this rate-distortion formulation depends on fixed cover image **x**, for simplicity of notation, **x** is not displayed in the equations. It has been proved in [1] that the optimal distribution $p(\mathbf{y})$ has the form of Gibbs distribution

$$p(\mathbf{y}) = \frac{\exp(-\lambda D(\mathbf{y}))}{\sum_{\mathbf{y}\in\mathcal{Y}} \exp(-\lambda D(\mathbf{y}))}, \tag{2.2}$$

where the parameter λ can be searched to achieve the average embedding rate of m bits. To facilitate implementation of embedding, the distortion function is approximated with a pixel-wise additive form [1, 2]

$$D(\mathbf{y}) \approx \sum_{i=1}^{n} d_i\left(y_i\right) \tag{2.3}$$

where $d_i(y_i)$ is the distortion for each pixel by changing x_i to y_i during embedding, while keeping all the other pixels in x unchanged, i.e., for $i \in \{1,...,n\}$, $d_i(y_i) = D(\mathbf{y})$, such that $\mathbf{y} = \{x_1,...,x_{i-1}, y_i, x_{i+1},...,x_n\}$. The additive approximation assumes independence between distortions caused by changing each one of the pixels in **x**. Under this assumption, according to Equation 2.2, the optimal probability distribution π_i for every pixel after embedding can be approximately by

$$p(\mathbf{y}) \approx \prod_{i=1}^{n} \pi_i(y_i) = \prod_{i=1}^{n} \frac{\exp(-\lambda d_i(y_i))}{\sum_{y_i \in \mathcal{I}_i} \exp(-\lambda d_i(y_i))} \tag{2.4}$$

where λ is searched to achieve $-\sum_{i=1}^{n} \sum_{y_i \in \mathcal{I}_i} \pi_i(y_i) \log_2 \pi_i(y_i) = m$ derived from Equation 2.1. Based on the information-theoretic framework formulated above, the design of $D(\mathbf{y})$ is the key issue left. Most of the researches in steganography under this formulation focus on the distortion design, e.g., all of the stegaonographic methods: HUGO [27], S-UNIWARD [3], HILL [34], and WOW [32] to be mentioned in the following text are characterized by their distinct definitions of the distortion functions.

This rate-distortion formulation enables simulation of embedding by changing each pixel i with π_i in Equation 2.4 to test the performance of the distortion functions. Note that this formulation is not the actual implementations of message embedding and extraction schemes. In practice, steganography with performance close to the additive rate-distortion bound could be realized with the syndrome-trellis coding [2]; but it is not the concern in this research.

Recall that the distortion function is used here to quantize the detectability (security) of steganography. The design of it often relies on heuristics and experience gained from steganalysis research. Numerous steganalysis works [11-14, 38, 39] have indicated that changes of pixel values in smooth regions are more detectable than changes made in high-frequency regions, i.e., textures, edges, etc. The reason is that steganalysis relies on statistics of pixel values; therefore, it is less accurate on modeling high-frequency regions which are less populated in images. This experience suggests higher embedding distortions in smooth regions and lower distortions in high-frequency regions. In our works, the S-UNIWARD (universal wavelet relative distortion in spatial domain) [3], a representative content-adaptive distortion function, is employed to test our steganalysis system. The S-UNIWARD distortion function is defined as

$$D(\mathbf{y}) = \sum_{k} \sum_{u,v} \frac{\left| w_{uv}^{(k)}(\mathbf{x}) - w_{uv}^{(k)}(\mathbf{y}) \right|}{\sigma + \left| w_{uv}^{(k)}(\mathbf{x}) \right|}. \tag{2.5}$$

Here the cover image **x** and the stego (data embedded) image **y** are treated as two dimensional. The $w_{uv}^{(k)}(\mathbf{x})$ and $w_{uv}^{(k)}(\mathbf{y})$ ($k \in \{1,2,3\}$) denote the first-level undecimated wavelet LH, HL, and HH directional decomposition of **x** and **y**, respectively, where *u* and *v* are the indices in the corresponding subband. Parameter σ serves as the stabilizing constant. The pixel-wise distortion $d_i(y_i)$ can then be derived from $D(y)$ one by one by changing x_i to $y_i \in \{x_i - 1, x_i, x_i + 1\}$, while keeping all the other pixels in *x* unchanged. Based on the denominator in Equation 2.5, the S-UNIWARD distortion assigns higher

Figure 2.1 An example of simulated embedding using S-UNIWARD with 0.4 bpp embedding rate. (Left) cover image. (Center) stego image. (Right) embedding changes.

distortion values to a pixel when its corresponding values in the high-frequency wavelet subband is lower; therefore, S-UNIWARD encourages embedding in high-frequency regions. More details can be found in [3].

Figure 2.1 gives a simulation example using S-UNIWARD[7] with 0.4 bit per pixel (bpp) embedding rate. The cover image is of size 512 × 512; hence, the payload (m) is 0.4 × 512 × 512 bits. Figure 2.1 (Left) shows the cover image and Figure 2.1 (Center) is the corresponding stego image, which is visually of no difference from the cover image. The embedding changes are displayed in Figure 2.1 (Right), in which the bright white pixels have been changed by +1 to the original pixel values in the cover image during embedding, the dark pixels have been changed by -1, and the gray pixels are unchanged. Although the embedding rate is 0.4 bpp, the actual change rate in this example is about 0.07 bpp, i.e., only 7% of the cover pixels values have been changed by either +1 or -1 to generate the stego image. We emphasize here that the changes made on the cover in Figure 2.1 (Right) are the actual signal of interest for steganalysis.

To summarize, advanced steganography has three strong points: (1) few changes are made on the least significant bits of cover pixels compared with the embedding rate;

[7] Source code available at http://dde.binghamton.edu/download/stego_algorithms/

(2) the locations of changes, depending on $(\pi_i)_{i=1}^n$ for simulation of embedding, are not fixed even with the same cover image; (3) changes strongly dependent on content of covers are made on locations less statistically detectable. Nevertheless, there are still two weak points: 1) the design of distortion functions in turn rely on heuristics and experience learned from steganalysis; 2) the additive assumption of distortions also compromises the optimality of data embedding.

2.1.2 Feature-based Steganalysis

To counter steganography, the goal of steganalysis is to identify if secret messages exist in given images. In this dissertation, steganalysis is treated as a binary (two-class) classification problem. Owning to the complex structures of natural images, classification with traditional methods in statistics that relies on accurate distribution modeling of images could hardly be applied. By contrast, machine learning (ML) methods skip the data modeling process and directly mine complex patterns with algorithmic models [4]. When combating with advanced steganography, ML-based steganalysis are generally more effective.

ML-based steganlaysis is data-driven and follows the general classification frameworks[8]. Let $\{\mathbf{x}_i, y_i\}_{i=1}^N$ denote an image dataset with *N* data, each have *n* pixels, we have $\mathbf{x}_i \in \mathcal{X} \subset \mathcal{I}^n$, where $\mathcal{X}$ is the data sampling subspace and $\mathcal{I} = \{0,...,255\}$ for 8-bit grayscale images; $y_i \in \mathcal{Y} = \{\text{cover}, \text{stego}\}$ is the corresponding label. For regression-based classifiers such as neural networks, $\mathcal{Y} = \{0,1\}$ is commonly used. Theoretically, we look for a mapping function $h: \mathcal{X} \to \mathcal{Y}$ using the given dataset so that the class labels of

[8] The notations in this section is not related to those used in 2.1.1.

newly observed data $\mathbf{x}^{new} \in \mathcal{X}$ will be predicted as $h(\mathbf{x}^{new})$. In practice, *h* is determined using part of the existing dataset, often called training set, so that the goodness of the selected *h* could be empirically evaluated by the error rate on the other part of the dataset, often called testing set. Denote the testing set as $\mathcal{T}$ and let the size of it be *M*, our goal is to select *h* to minimize the testing error:

$$\underset{h}{\text{minimize}} \frac{1}{M} \sum_{\mathbf{x}_i \in \mathcal{T}} I\left(h(\mathbf{x}_i) \neq y_i\right). \tag{2.6}$$

This error rate, under the assumptions of equal priors and same error costs for the two classes, is the major performance evaluation metric used in our steganalysis research.

Conventional ML-based steganalysis can be decomposed into two steps: feature extraction and pattern classification. The feature extraction step is applying a manually designed function *f* to transform every image data $\mathbf{x}_i$ into a *k*-dimensional real feature vector $\mathbf{z}_i$, namely, $\mathbf{z}_i = f(\mathbf{x}_i): \mathcal{X} \to \mathbb{R}^k$. The dataset is then transformed from $\{\mathbf{x}_i, y_i\}_{i=1}^N$ to $\{\mathbf{z}_i, y_i\}_{i=1}^N$. The feature vectors $\{\mathbf{z}_i\}_{i=1}^N$ serve as the inputs instead of $\{\mathbf{x}_i\}_{i=1}^N$ to some mathematically optimized (using the training set) generic classifier *g*, which maps the feature vectors to labels, namely, $g(\mathbf{z}_i): \mathbb{R}^k \to \mathcal{Y}$. Therefore, the mapping function *h* is a combined function of *f* and *g*: $h(\mathbf{x}_i) = g\left(f\left(\mathbf{x}_i\right)\right)$. While those well-developed generic classifiers such as support vector machines (SVMs) [5, 6] are the real strength of machine learning, their power could not be fully exerted without sophisticated feature extraction, particularly when the inputs are raw images pixels. In this dissertation, the machine

learning frameworks that heavily rely on manually designed features are called feature-based methods.

In feature-based steganalysis, given an image, the feature extraction procedure generally includes three essential steps: (1) generate the so-called noise residual images (called ‘residual’ in the following text) through high-pass filtering; (2) for each residual encode each pixel and its neighbors into a descriptor, and (3) statistically aggregate the descriptors to form final feature vectors.

Unlike most of the pattern recognition tasks, in steganalysis, the signals of interest are the embedding changes [Figure 2.1 (right)] mainly lying in the noise parts of cover image pixels, and the interference is the cover image content. In other words, steganalysis is a classification problem with extremely low signal-to-interference ratio (SIR). To boost the SIR, certain types of high-pass filtering are commonly performed in early stages of the feature extraction process to suppress the image content irrelevant to classification. In the literature, examples of high-pass filtering include but are not limited to taking high frequency wavelet subbands [7, 8], calibration in the JPEG domain [9, 10], and spatial mask filtering [11–14], etc.

The following descriptor generation and statistical aggregation steps that work on the residuals are inspired by the fact that the embedding changes made even in the least significant bits of pixel values would alter the complex dependencies of neighboring pixels. Typical methods in spatial domain work by obtaining either the high-order co-occurrence matrices of neighboring pixel values or histograms of random local linear projections on a rich and diverse set of noise residuals [12, 14].

The feature extraction of the SRM [12] (spatial rich model) is a good example that follows the typical procedure. Given an image, a rich and diverse set of residuals are first generated by mask filtering. Examples of the masks used are shown below:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0.5 & -1 & 0.5 \\ 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 0.5 & 0 \\ 0 & -1 & 0 \\ 0 & 0.5 & 0 \end{bmatrix} \quad \begin{bmatrix} -0.25 & 0.5 & -0.25 \\ 0.5 & -1 & 0.5 \\ -0.25 & 0.5 & -0.25 \end{bmatrix}$$

The residual generated from the above masks are called SPAM residuals, which could be further processed to generate MINMAX residuals by element-wise taking the minimum or maximum of the corresponding pixel values in multiple residuals. Next, the residuals, denote one of them as $\mathbf{r} = \left(r_{i,j} \right)$, will then be element-wise quantized and truncated with the equation

$$\hat{\mathbf{r}} = trunc(round\left(\frac{\mathbf{r}}{q} \right)) \tag{2.7}$$

where

$$trunc\left(a \right) = \begin{cases} T & a \geq T \\ -T & a \leq -T \\ a & otherwise \end{cases} .$$

The combined effect of truncation and quantization is essentially equivalent to performing binning to the residuals to facilitate the generation of histograms (or co-occurrence matrices) for accurate statistical modeling. Truncation also reduced the

interference caused by cover image content by limiting the large values in the residuals. Both the truncation threshold T and quantization q are empirically determined to be $T = 2$ and $q \in \{1,2,3\}$. Then, each element in $\hat{\mathbf{r}}$, denoted as $\hat{r}_{i,j}$, will be represented by a descriptor comprising the values of its four consecutive neighbors (including the pixel itself) in horizontal or vertical directions, followed by the statistically aggregation step of counting the co-occurrences of the descriptors across the whole residual map, e.g., for the horizontal 4-pixel neighborhoods, the co-occurrence values are calculated by

$$C(k_0,k_1,k_2,k_3) = \sum_{i,j} I\left(\hat{r}_{i,j} = k_0, \hat{r}_{i,j+1} = k_1, \hat{r}_{i,j+2} = k_2, \hat{r}_{i,j+3} = k_3\right) \tag{2.8}$$

where $k_0,k_1,k_2,k_3 \in \{-T,...,+T\}$. The formation of co-occurrence matrices discards the location information in one shot, thereby preventing the following generic classifiers to memorize the locations of embedding. To reduce the dimensionality and generate more concise and robust features, both the symmetric natures of co-occurrences and signs of residual values have been considered, i.e., $C(k_0,k_1,k_2,k_3)$, $C(k_3,k_2,k_1,k_0)$ and $C(-k_0,-k_1,-k_2,-k_3)$ are merged into one value, nevertheless, the feature dimensionality has still been boosted to more than 30,000 obtained from a total of 78 residuals, and yet, the classification performance is still not satisfactory. To further improve the steganalysis performance in face of the ever more sophisticated steganography algorithms, more discriminative statistical features are in demand. However, the manual feature design heavily relies on heuristics of steganalysis experts seems to have been pushed to the limit.

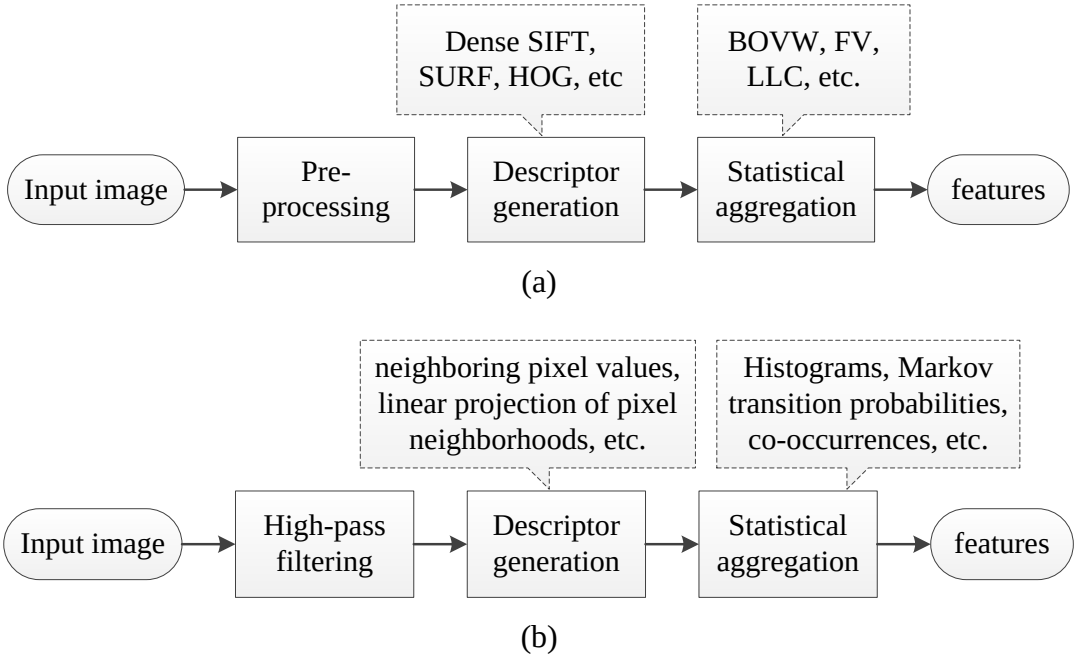


Figure 2.2 Feature extraction for (a) image classification and (b) steganalysis.

2.2 Motivation

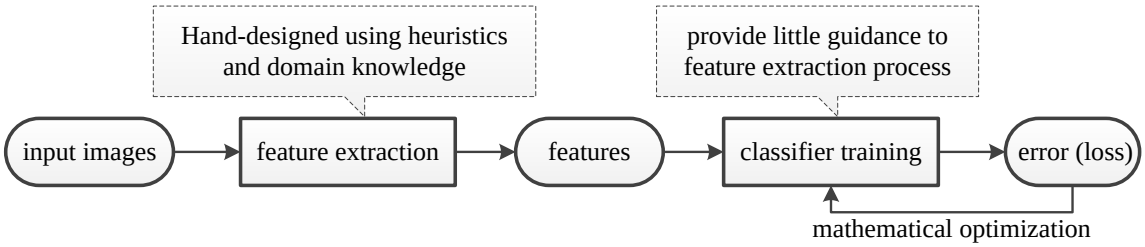
Steganalysis is not the only research field in which the feature extraction is difficult. In the field of image classification — determining what kind of object is presented in each image, one of the topics of broad interest in computer vision, feature extraction is also regarded as the key portion of the classification task and very difficult to improve. With a finite number of training samples, well-designed features must be robust to various cases of the objects, including scales, viewing angles, and occlusions; be robust to within-class diversities, e.g., different postures and breeds of animals; as well as be robust to complex backgrounds. Hence, the performance of classification with manual feature design would be far from optimal because of the aforementioned complexity. Popular approaches to feature extraction for image classification include a dense transform-invariant descriptor generation step to encode each pixel into a discriptor, e.g., scale-invariant feature

transform (SIFT) [15], speeded up robust features (SURF) [16], the histogram of oriented gradients (HOG) [17], followed by a statistical aggregation step on the descriptors, e.g., bag of visual words (BOVW) [18, 19], fisher vectors (FV) [20], locality-constrained linear encoding (LLC) [21]. The framework of feature extraction for image classification is summarized in Figure 2.2 (a); for comparison, the feature extraction for steganalysis is shown in Figure 2.2 (b). It is straightforward to realize that the feature extraction flow of steganalysis and that of image classification in computer vision are very similar.

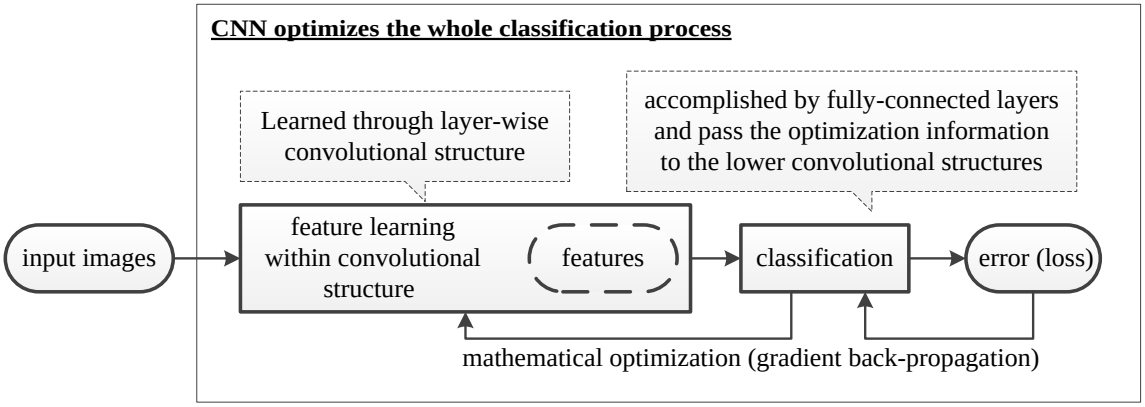
In year 2012, at the famous computer vision competition[9], a convolutional neural network (CNN) capable of learning features through mathematical optimization instead of manual feature extraction is designed by the winners [23] (main convolutional structure proposed early in the 1990s [22]). It surprisingly outperformed all the conventional feature-based methods adopted by other teams by a large margin [23, 24]. Since then, CNNs have been dominating in computer vision and an explosive amount of researches on CNNs are undergoing. In a short period of time, the routine to invest huge efforts for manual feature extraction has been replaced by the architectural design of CNNs capable of learning features.

The structure of CNN is characterized by its convolutional and pooling layers. In traditional neural networks, each of the output neuron in a hidden layer is fully connected with all the elements of the input. In contrast, in the convolutional layers, each of the output neuron is connected only within a predefined local region of input, and the parameters associated with the local connections are shared for all the output neurons. These constraints force the neural networks to focus on mining local spatial patterns, thereby capturing the essence of the input data — the strong spatial local-correlations in

[9] http://image-net.org/ (accessed on November 30, 2016)



(a)



(b)

Figure 2.3 Frameworks of feature extraction and classifier training for (a) feature-based and (b) CNN-based methods.

images. The other merit of the local connections and parameter sharing in convolutional layers is that the number of parameters to be optimized in the classifier is greatly reduced. The pooling layers aggregate local regions into more concise and informative representation. When the convolutional and pooling layers are alternately placed, the CNN learns optimized hierarchical features through gradients back-propagation [25]. Figure 2.3 gives a comparison of the traditional feature-based framework and the CNN-based framework. More details of the layer functions will be introduced along with the presentation of our designed CNN architecture in Section 2.3.

Due to the similarity between the classification frameworks of steganalysis and image classification, and their same difficulty in feature design, we believe that the CNNs could potentially learn more effective features to boost the performance of steganalysis.

2.3 Architectural Design of CNN

The success of CNN in computer vision and the similarity of classification frameworks of steganalysis and image classification have aroused the interest of us in seeking the way to use CNNs for steganalysis. Nevertheless, recent studies conducted by other researchers [26, 37] have indicated that the architectures of CNNs tailored for computer vision may not be best suited to image steganalysis. After all, image classification and steganalysis are different research topics. Some comparative analysis is necessary to understand the difference.

For any classification problems, we need to be clear what the signal of interest is and what the noise or interference is. The task of image classification is to recognize the type of major objects, which are part of image content and the signals of interest; the interference is the rest of image content, e.g., backgrounds. In contrast, the signals of interest in steganalysis are the changes made to pixels, e.g., ±1 to pixel values for ternary embedding [see Figure 2.1 (Right)]; the interference is the content of cover images. According to steganography introduced in Section 2.1.1, the signal and interference are very dependent and the signal-to-interference ratio (SIR) is extremely low. The within-class diversities for classification are also different. For image classification, possible within-class diversities include variations of scales, viewing angles, spatial locations in images, occlusions, etc. For steganalysis, the variation of image content is a within-class difficulty for both the cover and stego classes; the unfixed embedding locations and changes of pixel values even with fixed covers is another difficulty for the stego class.

Aware of the difference between steganalysis and image classification, successful CNN architecture for steganalysis should (1) enhance the SIR, (2) weaken the

interference brought by the content of cover images, (3) prevent memorizing the exact locations of embedding changes from the training set, and (4) learn from feature extraction in traditional feature-based steganalysis.

Before the publications of our works, two pieces of works have been published by other researchers in this field. In their pioneering work, Tan and Li [26] proposed a CNN which comprises three stages of alternating convolutional layers with sigmoid non-linear activations, and max-pooling layers with stride equals 4 (4×4 down-sampling). When detecting HUGO [27], which is an earlier version of content-adaptive steganography, at embedding rate of 0.4 bpp on the BOSSbase [28], the CNN had an error rate of 48% with random parameter initialization; after involving a high-pass convolutional kernel [12, 27] into parameter initialization of the first convolutional layer, and pre-training all of the parameters with unsupervised learning, they managed to reduce the error rate to 31%, still far away from the 14% achieved by the SRM [12] and FLD-ensemble (fisher linear discriminants as weaker leaners) [30]. The major weaknesses of their proposed CNN are the max-pooling operation, which relies heavily on the image content; the stride-4 down-sampling rate (in contrast to stride-2 commonly used) during pooling, which causes too much information loss; and the 5×5 convolution kernel size in deeper convolutional layers that may overly model pixel neighborhoods.

A few months later, Qian et al. [31] reported a CNN equipped with a high-pass filtering layer, Gaussian non-linear activations, and average pooling for steganalysis. The reported detection error rates are 2% to 5% higher than those achieved by the SRM on the BOSSbase when detecting three content-adaptive steganography — HUGO [27], WOW [32], and S-UNIWARD [3]. This is a significant boost in performance, but it is still

inferior to the SRM. The improvements could be mainly attributed to the average-pooling and the high-pass filtering layer adopted. However, it is rather difficult to reproduce the reported results due to the multi-layer stack of Gaussian activation which makes training extremely difficult without proper initialization.

Studies in these two pieces of works have indicated that taking into account the domain knowledge in steganography and steganalysis, e.g., using high-pass kernel to generate noise residuals, improves the classification performance of the CNNs. In feature-based steganalysis, the domain knowledge is embedded in the manual feature extraction step. Analogously, as the CNNs embrace the feature extraction step into the networks, the domain knowledge should be reflected in the network architectures.

Along this direction, we propose a CNN that tries to incorporate the knowledge of steganalysis. In the detailed architecture, we take absolute values of elements in the feature maps generated from the first convolutional layer to facilitate and improve statistical modeling in the subsequent layers; to weaken the interference caused by image content, we constrain the range of element values at early stages of the networks; to prevent overfitting, the strength of modeling is reduced by using 1×1 convolutions in deeper layers; besides, as have been proved effective in the previous works [26, 31], the proposed CNN learns from noise residuals to improve the SIR and uses average pooling.

Although the proposed CNN is neither large nor deep, and currently learns from only one type of noise residual, the results have verified that its performance is comparable with that of the SRM. This initial-stage work has confirmed that deep learning with CNNs is indeed a powerful machine learning tool for steganalysis. The

results have also implied that a well-designed CNN would have the potential to provide a better detection performance compared with the traditional feature-based steganalysis.

In this section, first, the overall architecture of the proposed CNN is directly given in Section 2.3.1. Then, in Sections 2.3.2, 2.3.3, and 2.3.4, we discuss about our design considerations. All the experimental results to support the design appeared in Section 2.3.2, 2.3.3, and 2.3.4 were obtained using cross-validation on the training set. Details of the dataset, software platforms, data splits, and hyper-parameters involved in the CNN are covered in Section 2.3.5. Results on the testing set are presented in Section 2.3.6. Conclusions are drawn in Section 2.3.7.

2.3.1 Overall Architecture

In this section, the entire layer functions involved in our proposed CNN are elaborated. These layer functions constitute the forward function of the entire CNN, enough for understanding the ideas of design for steganalysis; the optimization (training) of the CNN is enabled by gradient back-propagation [25]. We would like to emphasize that our contribution is the whole architectural design for steganalysis, not the layer functions as components in the CNN.

Figure 2.4 illustrates the overall architecture of our CNN. Inside boxes are the layer functions and hyperparemeters. Data sizes are displayed on the two sides. Sizes of convolution kernels in the boxes follow (number of kernels) × (height × width × number of input feature maps). Sizes of data follow (number of feature maps) × (height × width).

Same as in [31], a high-pass filtering (HPF) layer is placed at the very beginning to transform original images to noise residuals ($\mathcal{I}^{512\times512} \rightarrow \mathbb{R}^{512\times512}$, the input images are

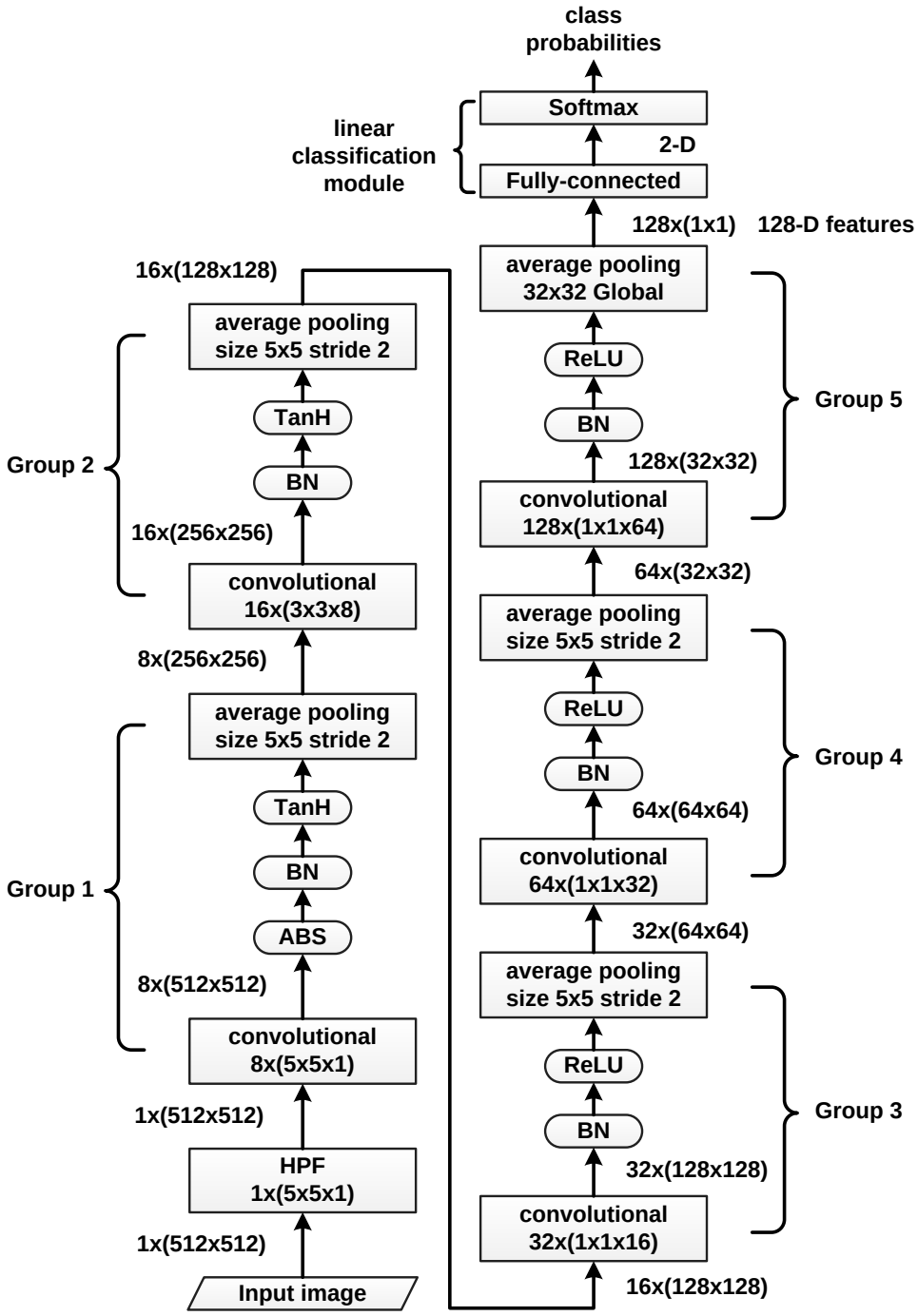


Figure 2.4 The proposed CNN architecture.

of size 512×512)[10] in order to boost the SIR. The HPF kernel is one that commonly used in steganalysis research shown below:

$$W = \frac{1}{12}\begin{bmatrix} -1 & +2 & -2 & +2 & -1 \\ +2 & -6 & +8 & -6 & +2 \\ -2 & +8 & -12 & +8 & -2 \\ +2 & -6 & +8 & -6 & +2 \\ -1 & +2 & -2 & +2 & -1 \end{bmatrix} \tag{2.9}$$

The parameters in this 5×5 kernel are fixed and not optimized during training. Therefore, the actual inputs to the CNN are noise residuals, not the original images.

The whole CNN can be divided into a convolutional module followed by a linear classification module. The convolutional module transforms the noise residuals to 128-dimensional (128-D) feature vectors. The linear classification module, equivalent to logistic regression for two-class classification problem, composed of a fully-connected (FC) layer and a softmax layer, and routinely transforms the feature vectors to posterior probabilities for each class. Final class labels are determined by choosing the class corresponding to the larger posterior. In this work, we focus on the design of the convolutional module.

The convolutional module comprises five groups of layers (displayed as Group-1 to Group-5 in Figure 2.4), each starts with a convolutional layer which generates feature maps, and ends with an average pooling layer which performs local averaging (except Group 5) as well as subsampling on the feature maps.

[10] Throughout the presentation in this section, we always assume proper padding is applied wherever is necessary.

Both the inputs and outputs of convolutional layers are three-dimensional (except the convolutional layer in Group-1). Let the input to a convolutional layer be of size $H \times W \times C$, where H and W are the sizes of two spatial dimensions and C is the number of input feature maps (sometimes called channels), the output of this convolutional layer has size $H \times W \times K$, where the two spatial dimensions are same as those of the input and the output has K feature maps. The convolutional kernels containing parameters are of size $M_H \times M_W \times C$, where M_H and M_W ($M_H \leq H$, $M_W \leq W$) are the spatial sizes[11] of the kernels and the third dimension equals the number of input feature maps. Functionally, sliding window dot-product is first performed across spatial dimensions of the input datum with the kernel so that C maps are generated, which are then element-wise summed; therefore, one output feature map of size $H \times W$ can be generated by a single kernel. To generate K output feature maps, K kernels of size $M_H \times M_W \times C$ are needed in the convolutional layer. For example, the convolutional layer in Group-2 of Figure 2.4 has input size of 256×256×8 ($H = W = 256$, $C = 8$), output size of 256×256×16 ($K = 16$); therefore, there are 16 kernels of size 3×3×8 ($M_H = M_W = 3$) in this convolutional layer. Apart from the convolutional kernels, each output feature map is element-wise added by a single bias value. Values in the kernels as well as biases are the parameters in the convolutional layer to be optimized. Let $\mathbf{x} = \left(x_{i,j,c} \right)$ be the three-dimensional input to the convolutional layer, where $i \in \{1,...,H\}$ and $j \in \{1,...,W\}$ are the spatial indices in the c-th ($1 \leq c \leq C$) feature map, and let $\mathbf{w}^{(k)} = \left(w_{u,v,c}^{(k)} \right)$ be the k-th ($1 \leq k \leq K$) kernel, where $u \in \{1,...,M_H\}$ and $v \in \{1,...,M_W\}$ are its spatial indices and c corresponds to the same feature map as the

[11] Assume both M_H and M_W are odd numbers.

input datum, and further let $b^{(k)}$ be the bias element-wise added to the *k*-th output feature map[12], the corresponding output element $y_{i,j,c}^{(k)}$ of the convolutional layer is calculated by

$$y_{i,j,c}^{(k)} = b^{(k)} + \sum_{c=1}^{C} \sum_{u=1}^{M_H} \sum_{v=1}^{M_W} w_{u,v,c}^{(k)} x_{i-\lceil M_H/2 \rceil + u,\, j-\lceil M_W/2 \rceil + v,c} \tag{2.10}$$

The corresponding matrix form is

$$\mathbf{y}^{(k)} = b^{(k)} + \sum_{c=1}^{C} \mathbf{w}_c^{(k)} * \mathbf{x}_c \,, \tag{2.11}$$

where $\mathbf{x}_c$ is the *c*-th input feature map of size $H \times W$, $\mathbf{w}_c^{(k)}$ of size $M_H \times M_W$ is the 2-D mask applied on $\mathbf{x}_c$ in the *k*-th kernel, $\mathbf{y}^{(k)}$ of size $H \times W$ is the *k*-th output, and the operator '$*$' denotes the usual spatial convolution in image processing.

To enhance the power of statistical modeling, our CNN is equipped with the hyperbolic tangent (*TanH*) [Figure 2.5 (left)] non-linear activations for Group-1 and Group-2, and the rectified linear unit (*ReLU*) [35] activations [Figure 2.5 (right)] for Group-3, Group-4, and Group-5. Inside Group-1, an absolute activation (ABS) layer is inserted to force the statistical modeling to take into account the (sign) symmetry [12][14] existed in noise residuals. To prevent the CNN training from falling into poor local minima, immediately before each non-linear activation layer, the feature maps are normalized with batch-normalization (BN) [36].

[12] Biases are fixed to be zeros in the Group 1 which will be covered in section 2.3.2.

The BN layer first normalizes elements in each feature map of the input to zero-mean and unit-variance to ensure that the initial input to the following *TanH* activations falls in the quasi-linear region, as shown in Figure 2.5 (Left), so that the gradient back-propagation would not fall into poor local minima. Unlike the other layer types, the BN layer only works when there are more than one data presented. The input to a BN layer should have N ($N > 1$) data, each have dimensions $H \times W \times K$. Let n denotes the n-th input data ($1 \le n \le N$), k denotes the k-th feature map ($1 \le k \le K$), and let i and j ($1 \le i \le H$, $1 \le j \le W$) be the spatial indices in the feature maps, the normalization for input elements in the k-th feature maps of the n-th data can be written as

$$\overset{(k)}{x}_{n,i,j} = \frac{x_{n,i,j}^{(k)} - \mu^{(k)}}{\sigma^{(k)}} \tag{2.12}$$

where $\mu^{(k)} = \frac{1}{NHW} \sum_{n,i,j} x_{n,i,j}^{(k)}$ and $\sigma^{(k)} = \sqrt{\frac{1}{NHW} \sum_{n,i,j} \left(x_{n,i,j}^{(k)} - \mu^{(k)} \right)^2}$. To recover the power of modeling, the BN layer then scales and shifts the normalized data with a scaling factor $\gamma^{(k)}$ and a bias $\beta^{(k)}$, both optimized, for each normalized feature map, and generates an output element of $y_{n,i,j}^{(k)}$ which can be calculated by

$$y_{n,i,j}^{(k)} = \gamma^{(k)} \overset{(k)}{x}_{n,i,j} + \beta^{(k)} \tag{2.13}$$

Since all the normalization, scaling and shifting processing in the BN layer are identical within each feature map and differ across feature maps, the spatial correlations

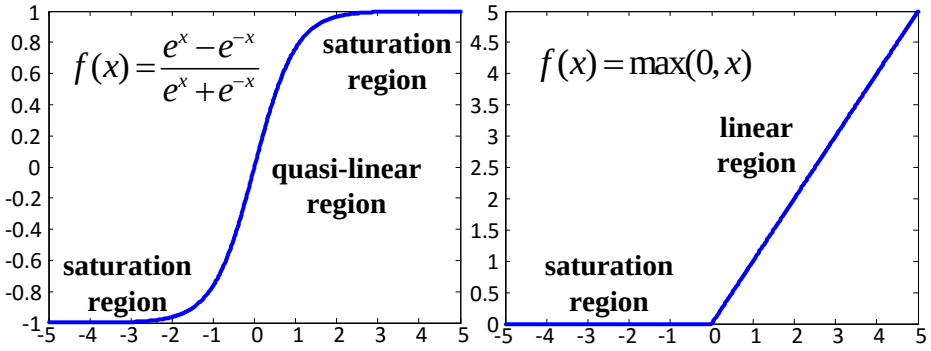


Figure 2.5 The two non-linear activation functions. (Left) *TanH*. (Right) *ReLU*.

within feature maps are well preserved, and optimal scaling and shifting parameters could be learned for each feature map. The output feature maps from the BN layer are then element-wise mapped by the *TanH* activation into three regions, as shown in Figure 2.5 (Left), and the output are well-prepared for further modeling.

The pooling layers in Group 1–4 of Figure 2.4 perform local averaging on every other input element (stride = 2) in the spatial dimensions, for each input feature map independently. Therefore, the outputs of the pooling layers have the same number of feature maps and are of half the sizes in the two spatial dimensions, i.e., input of size $H\times W\times K$ are reduced to $(H/2)\times(W/2)\times K$ after pooling. Let the spatial region sizes for averaging be $M_H\times M_W$ ($M_H = M_W = 5$ in our work), output elements of the pooling layers can be expressed as

$$y_{i,j}^{(k)} = \frac{1}{M_H M_W}\sum_{u=1}^{M_H}\sum_{v=1}^{M_W} x_{2i-\lceil M_H/2\rceil+u,\,2j-\lceil M_W/2\rceil+v}^{(k)} \tag{2.14}$$

Finally, through global averaging, the pooling layer in Group 5 merges each spatial map to a single element (128 feature maps of size 32 × 32 to a 128-D feature vector), i.e.,

$$y^{(k)} = \frac{1}{HW} \sum_{i,j} x_{i,j}^{(k)} \tag{2.15}$$

where $H = W = 32$ and $k \in \{1,...,128\}$ according to Figure 2.4. In this way, the whole CNN is constraint to perform the same operations to every pixel in the original images (or the noise residuals), thereby preventing the statistical modeling from grasping the location information of embedded pixels from the training data.

2.3.2 Layer Designs for Statistical Modeling

The exact modeling procedure in the CNN is hard to interpret when the layer-wise computation goes deeply. Therefore, we stand a better chance to improve the performance by focusing more on the design of the first layer group (Group-1 in Figure 2.4), where the functionality of the CNN is still traceable.

Group-1 starts with a convolutional layer that takes as input the noise residuals generated from the HPF layer. This convolutional layer explores relations of neighboring pixels in the residuals with optimized kernels $\left(\mathbf{w}^{(k)}\right)_{k=1}^{K}$, and generates feature maps for statistical modeling. Unlike the three-dimensional input of the other convolutional layers, since the HPF layer only generates one residual for each data, the input to the convolutional layer in Group-1, denoted as $\mathbf{x}$, is two-dimensional, i.e., the number of

input feature map C equals 1. To assist the statistical modeling of the CNN, we disable the default bias learning in this convolutional layer, namely, the biases $b^{(k)}$ ($k \in \{1,...,K\}$) in Equations 2.10 and 2.11 are forced to be zeros, so that the output feature maps are symmetric with respect to zeros[13].Thus, according to Equation 2.11, for $k \in \{1,...,K\}$, the convolutional layer in Group-1 has a simplified function:

$$\mathbf{y}^{(k)} = \mathbf{w}^{(k)} * \mathbf{x} \tag{2.16}$$

Then, we insert an ABS layer right after this convolutional layer to discard the signs of the elements in the feature maps, denoted as $\left|\mathbf{y}^{(k)}\right|$, $k \in \{1,...,K\}$, where $|...|$ stands for taking element-wise absolute values. The output of the ABS layer is thereafter fed into a BN layer, which performs optimized scaling and shifting on each of the $\left|\mathbf{y}^{(k)}\right|$ for $k \in \{1,...,K\}$. The output of the BN layer would then be element-wise truncated by the saturation regions [see Figure 2.5 (Left)] in the following *TanH* activation function. Denote $\mathbf{z}^{(k)}$ as the corresponding output map of the *TanH* function, the joint function of the BN layer and *TanH* activation is

$$\mathbf{z}^{(k)} = TanH\left(m^{(k)}\left|\mathbf{y}^{(k)}\right| + n^{(k)}\right) \tag{2.17}$$

[13] The sign-symmetry [12][14] is brought by natural image statistics and the equally treated ±values in the distortion function of steganography (see Equation 2.5).

where $m^{(k)} = \frac{\gamma^{(k)}}{\sigma^{(k)}}$ and $n^{(k)} = \beta^{(k)} - \frac{\gamma^{(k)}\mu^{(k)}}{\sigma^{(k)}}$ according to Equations 2.12 and 2.13 are the optimized scale and bias for the *k*-th output feature map. In fact, the *TanH* function can be considered as an approximation of the truncation function introduced in Equation 2.7,

$$TanH(a) \approx trunc(a) = \begin{cases} T & a \geq T \\ -T & a \leq -T \\ a & otherwise \end{cases} \tag{2.18}$$

The functionality of Equations 2.17 and 2.18 bears some similarity with the quantization and truncation steps in the SRM feature extraction. Some minor difference is (1) the hard quantization and truncation in SRM (Equation 2.7) is replaced by the softer scaling → shifting → *TanH* operation chain, with both the scaling and shifting values mathematically optimizable; (2) encoding of pixel neighborhood happens earlier by linear projection with optimized kernels $\left(\mathbf{w}^{(k)}\right)_{k=1}^{K}$, whereas in SRM the encoding is performed later by considering the four neighbors during generation of the co-occurrence matrices. Note that the second difference point is more similar to another feature-based method called PSRM [14], in which the projection kernels are not optimized but randomly generated. The more different here is the following statistical aggregation step, which, in the SRM, is achieved with a one-shot generation of co-occurrence matrices, whereas in the CNN, is performed by layer-wise stacking of convolutional and pooling layers, and is therefore, expected to be more powerful, but in the meantime, much more difficult to interpret.

To obtain some intuitive understanding of the functionality, we drawn in Figure 2.6 the distributions of a validation image after it went through the convolutional layer, ABS layer, BN layer, and *TanH* activation in Group 1 of a trained CNN (only the first two feature maps are displayed). As we disabled bias learning in the first convolutional layer, the output distributions, in Figure 2.6 (a), are symmetric with respect to zeros. The outputs of the ABS layer, with the distributions shown in Figure 2.6 (b), are first normalized following Equation 2.13 (with the global statistics stored during training). The normalized feature maps are then scaled and shifted (with bias) following Equation 2.11 for optimal statistical modeling, the distributions after this step are displayed in Figure 2.6 (c). Table 2.1 records the optimized scaling and shifting values for all the eight feature maps in the first BN layer. The output of the BN layer are then mapped and bounded by the *TanH* activation, as shown in Figure 2.6 (d).

Because of the bias terms introduced in the BN layer, without the ABS layer, the sign-symmetry of elements in the output feature maps of the first convolutional layer would no longer hold, causing interference between feature values at early stages of statistical modeling. Recall that in the SRM feature extraction, this symmetry is used by merging the bins in co-occurrence matrices. However, in the CNN, the statistical aggregation step is replaced by hierarchical local convolution and pooling in the deeper layers; there is nowhere else suitable to inject this symmetry. This problem could be solved once the sign information is discarded by the ABS layer. In Figure 2.7, both the training and validation results are reported along with the iterative training processes with a five-fold cross-validation on the training set. We observe worse results for both training and validation once the ABS layer is removed from the proposed CNN.

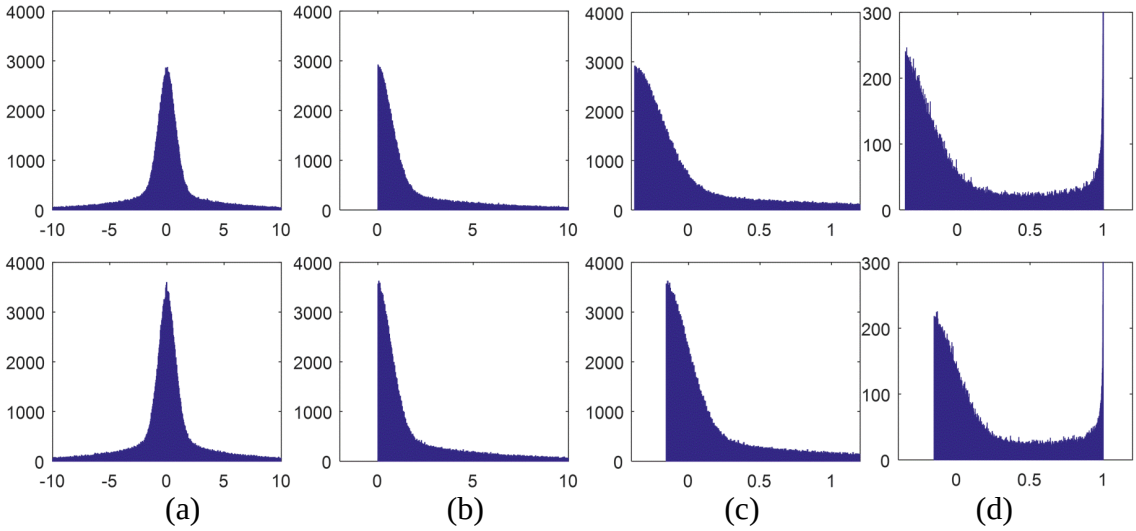


Figure 2.6 Distributions of the first feature map (first row) and second feature map (second row) after going through (a) convolutional layer, (b) ABS layer, (c) BN layer, (d) TanH layer, in Group-1.

Table 2.1 Optimized Scales and Biases in The BN Layer after The Normalization Step

k	1	2	3	4	5	6	7	8
γ^k	1.20	1.24	1.15	1.61	0.91	0.93	1.04	1.10
β^k	0.29	0.51	0.50	0.55	0.25	0.33	0.58	0.40

The other feature in our proposed CNN is the use of *TanH* non-linear activation function rarely seen in modern neural networks, which, almost exclusively use *ReLU* [23, 24, 36, 45, 46]. The major reason is that the saturation regions on the two sides of *TanH* make gradient back-propagation less efficient in deep neural networks, while the *ReLU* with only one side saturated does not have such a problem. Nevertheless, the *TanH* activations miraculously return in our CNN, and have made two significant contributions: (1) they provide efficient truncation function for statistical modeling as mentioned earlier; (2) they effectively limit the range of data values [illustrated in Figure 2.5 (Left) and Figure 2.6 (d)] and prevent the deeper layers from modeling large values more related to

image content. In fact, in our CNN designed for steganalysis, a hybrid of *TanH* and *ReLU* non-linear activations are employed to embrace the strong points in both of them. Figure 2.8 compares the results when the two *TanH* activations in Group-1 and Group-2 of the proposed CNN are replaced by the *ReLU*s. Again, we observe performance drop on both the training and validation set. However, it has also been discovered that results became worse when more ReLUs in deeper layers were replaced by the *TanH*, likely due to the difficulty of gradient back-propagation with *TanH* [37]. More results are provided in the Appendix.

2.3.3 Constraining the Power of Modeling

In this section, we describe our considerations on some other parts of the CNN design, including spatial sizes of convolutional kernels, and selection of pooling types.

Traditional feature-based steganalysis models patterns of pixel correlations in a small local region of the residual maps, and adopts one-shot histogram pooling or co-occurrence pooling to prevent modeling larger regions [11-14, 38, 39]. In contrast, the CNN works by modeling (with alternating convolutional and pooling layers) relationships of residual elements over the whole images. On the one hand, such type of modeling is one of the root strengths of the CNN, on the other hand, without effective control, overfitting (fitting the stego noise in training data too well to generalize to testing data) could occur for steganalysis. Having realized this potential issue, in the proposed CNN, we limit the sizes of convolutional kernels in deeper layers, i.e., the spatial sizes of the convolutional layers in the last three groups are limited to 1×1 so as to constrain the strength of modeling. The function in convolutional layer is simplified to Equation 2.19 when the kernel size become 1×1×*C*, i.e., the 1×1 convolution in essence gives up the

convolution operation for modeling spatial relations in feature maps in the deeper layers, but only perform dot product across feature maps.

$$\mathbf{y}^{(k)} = b^{(k)} + \sum_{c=1}^{C} \mathbf{w}_c^{(k)} \mathbf{x}_c \tag{2.19}$$

Figure 2.9 presents the training and validation errors when replacing the 1×1 convolutions with 3×3 and 5×5 convolutions. It is observed that with the growth of the spatial dimensions of the kernels, training errors reduces significantly but validation errors are just slightly worse, which indicates fitting training data too well so that the generalization suffer. Note that the selection of 1×1 convolution should not be taken for granted. It is expected that with more training data larger convolutional sizes would be preferred.

Same as proposed in [31], the proposed CNN also favors average pooling over max pooling, which output the maximum value of a local region compared with the mean value as shown in Equation 2.14, or convolution with strides, which performs regular convolution spatially with step size equals 2 [40]. Our understanding is that, in essence, max pooling is a competing method within the pooling region, therefore, the output of max pooling depends heavily on image content as well as the locations of stego noise in training data, which would harm the generalization ability of the CNN. In contrast, average pooling aggregates information by low-pass filtering with fixed kernels, and hence is conservative in terms of modeling and less likely to overfit. Figure 2.10 clearly

demonstrate the benefit of using average pooling compared with the other two possible competitors.

2.3.4 Additional Cross-Validation Results

In this section, some additional cross-validation results to support the CNN design in Section 2.3 are presented.

In Section 2.3.1, we mentioned that the HPF layer was initialized with a sophisticated 5×5 kernel whose parameters were not updated during training. Figure 2.11 shows that without this HPF layer, the training errors does not decrease, which indicates unsuccessful learning caused by the overwhelming interference of cover content (extremely low SIR). Therefore, adding the HPF layer generating residuals to boost SIR would be an essential move towards success. If the parameters in HPF kernel are initialized with the sophisticated kernel but also being updated during the training process, as shown in Figure 2.12, the CNN experienced difficulty in convergence during roughly half of the earlier iterations of training, and then converged to a worse result compared with the CNN with the HPF parameters fixed. Most likely, the unoptimized parameters in the other parts of the CNN caused ‘incorrect’ updates of the well-initialized HPF kernels, which in return produced ‘noisy’ residuals made the training too noisy to converge; this problem disappeared after all of the parameters in the CNN were relatively optimized, yet the optimization results were still worse.

Figures 2.13 and 2.14 demonstrates the selection of pooling sizes and non-linear activation functions, respectively.

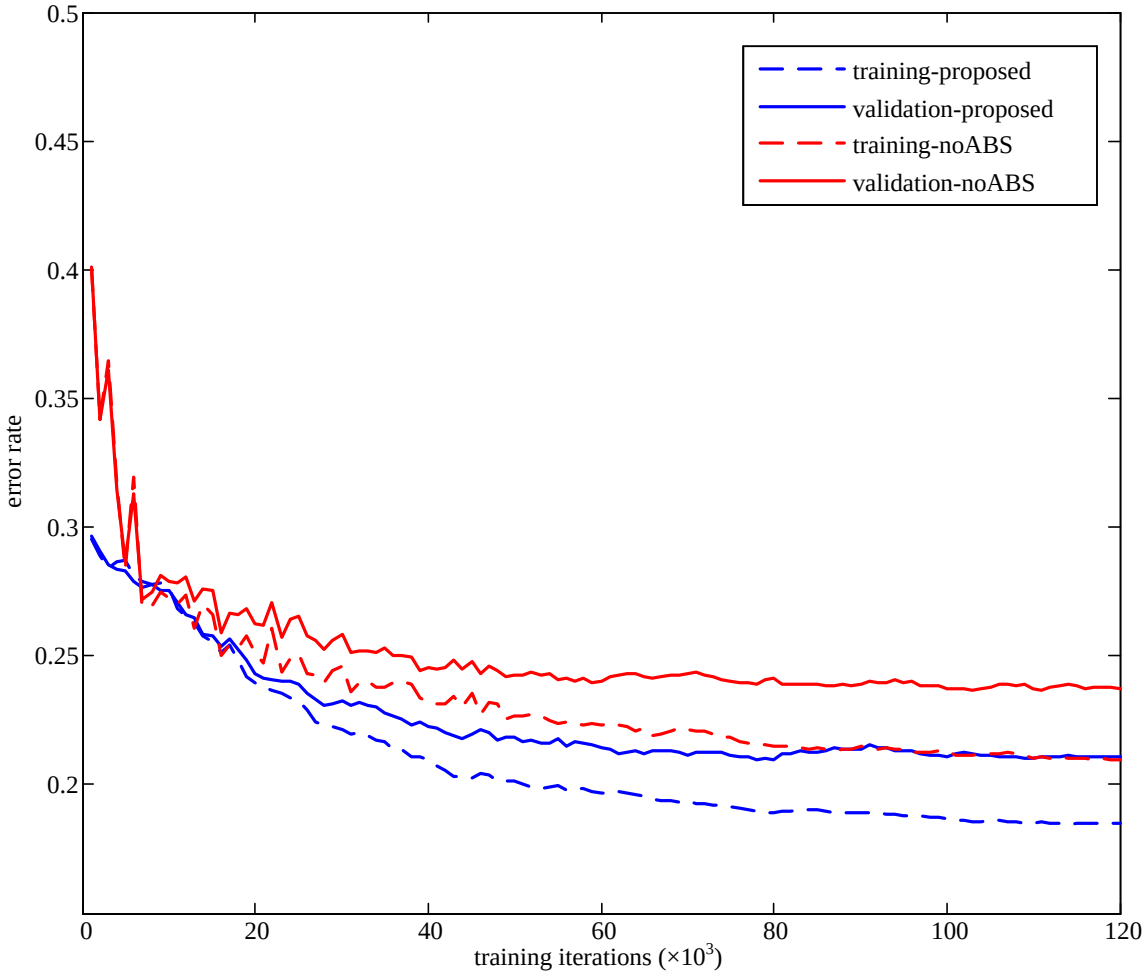


Figure 2.7 Training errors and validation errors: proposed CNN vs. the CNN without the ABS layer.

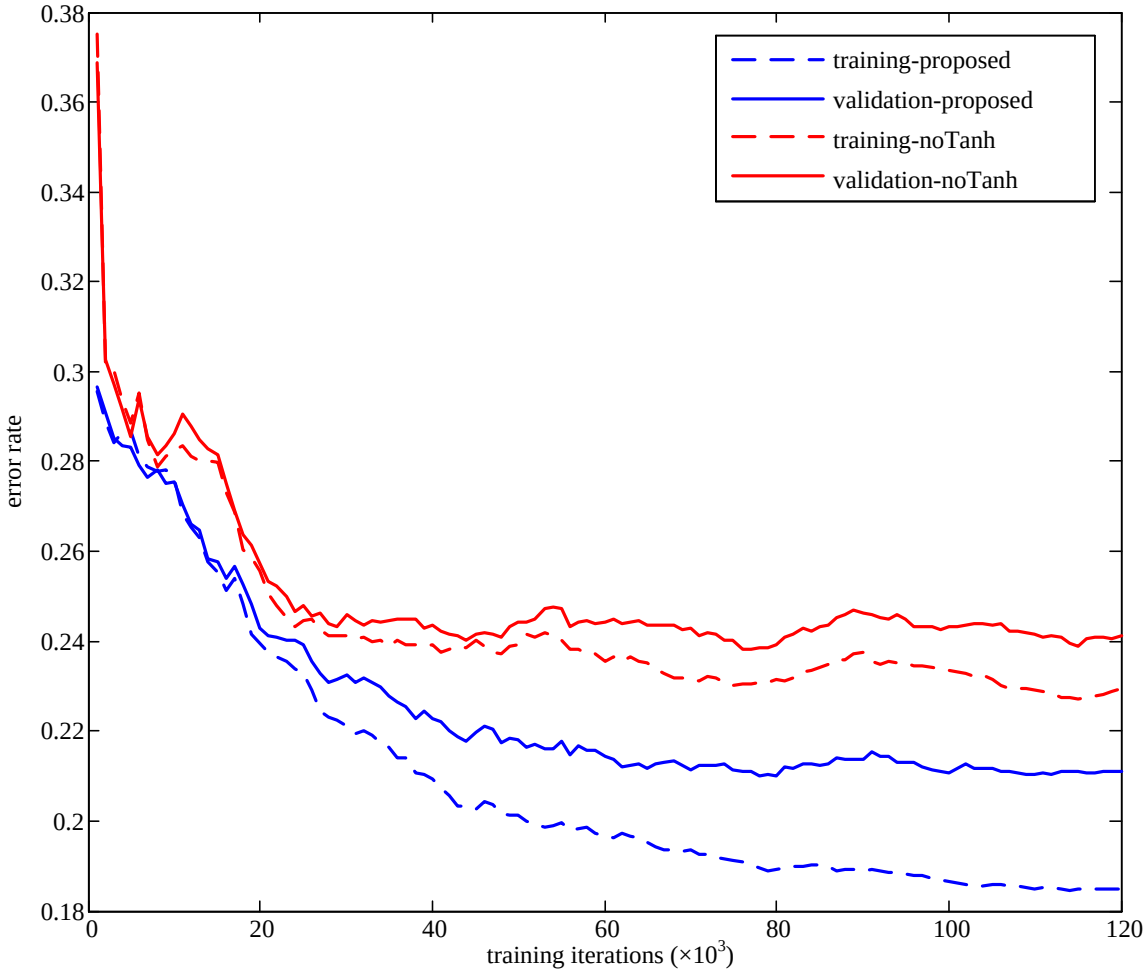


Figure 2.8 Training errors and validation errors: proposed CNN vs. the CNN replacing *TanH* with *ReLU* in Group-1 and Group-2.

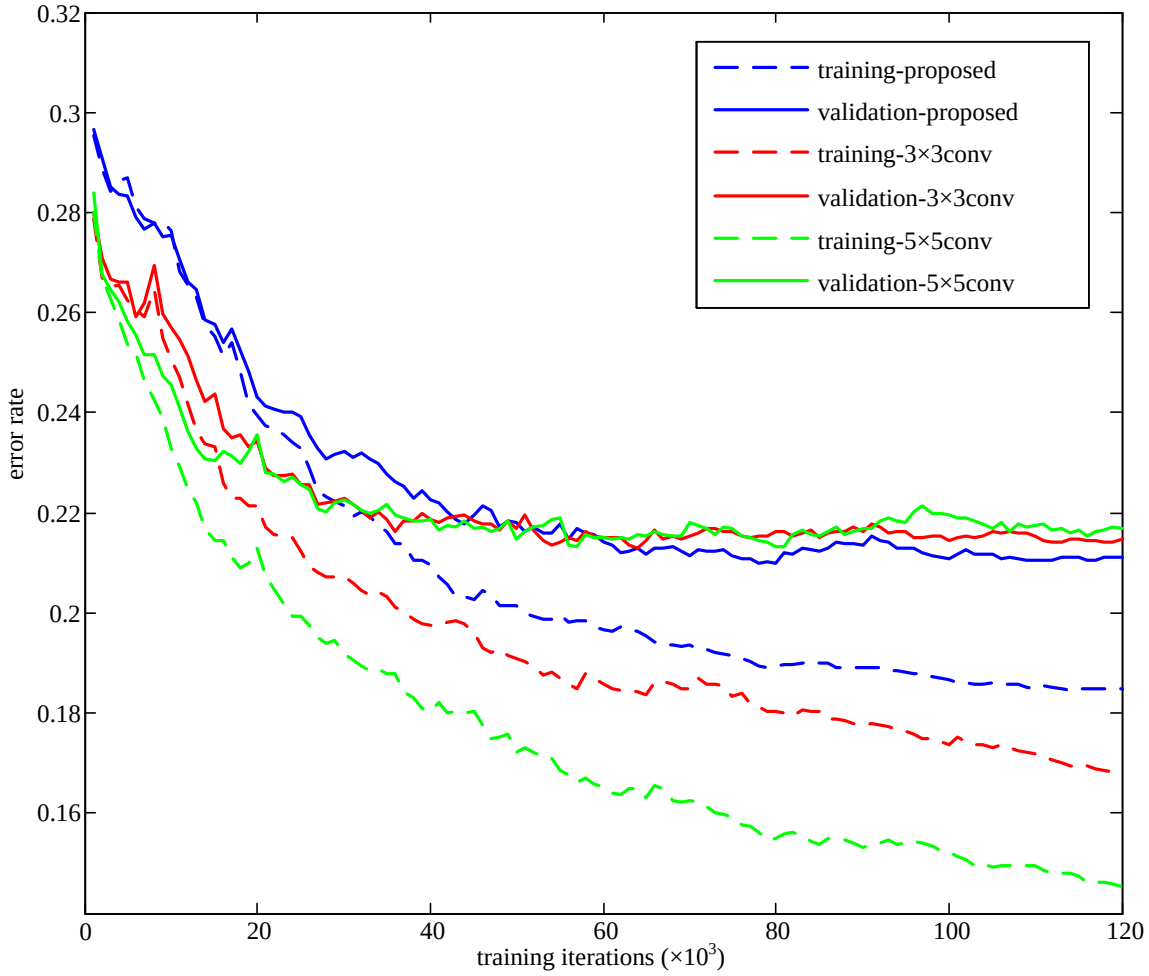


Figure 2.9 Training errors and validation errors: proposed CNN vs. the CNNs replacing 1×1 convolutions with 3×3 and 5×5 convolutions in Group-3 – Group-5.

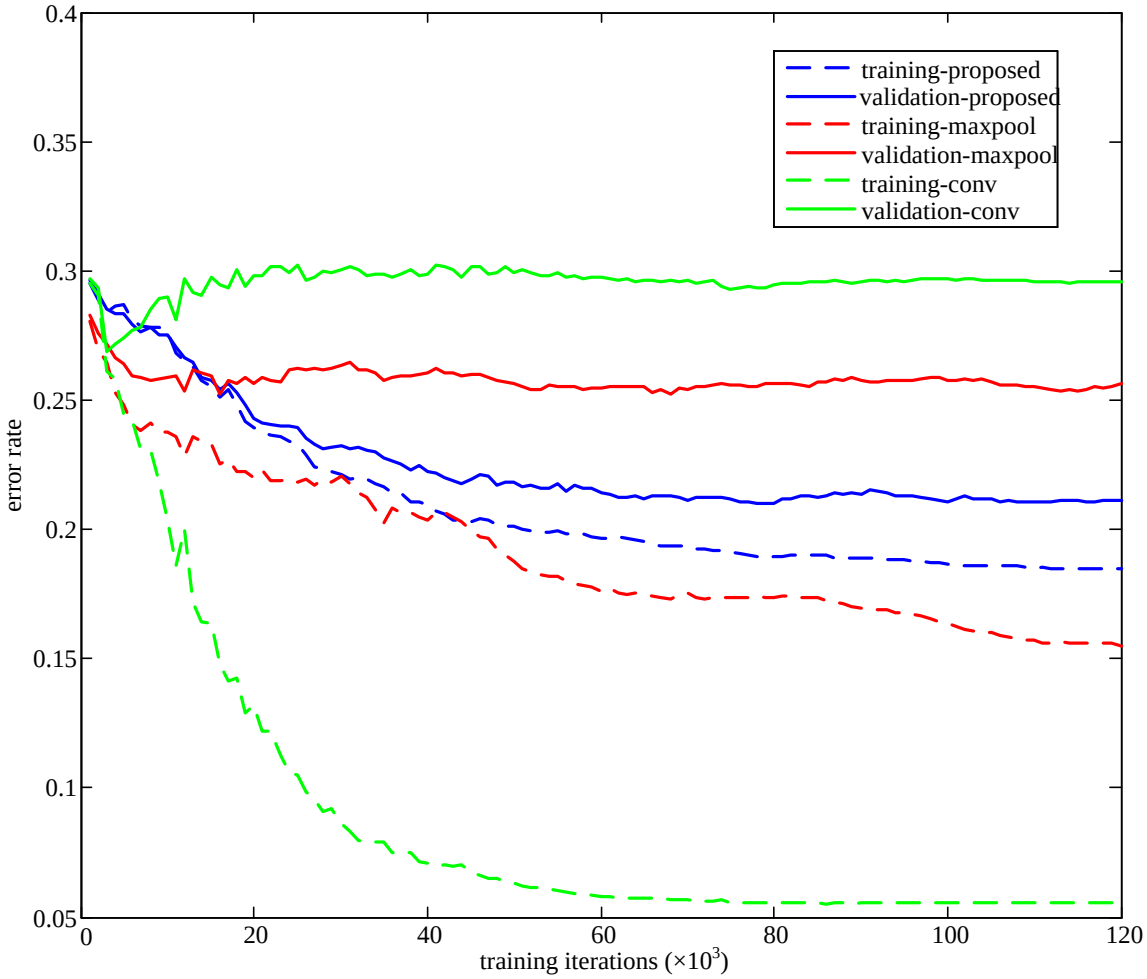


Figure 2.10 Training errors and validation errors: proposed CNN vs. the CNNs replacing average-pooling with max-pooling and convolutional layers with stride = 2.

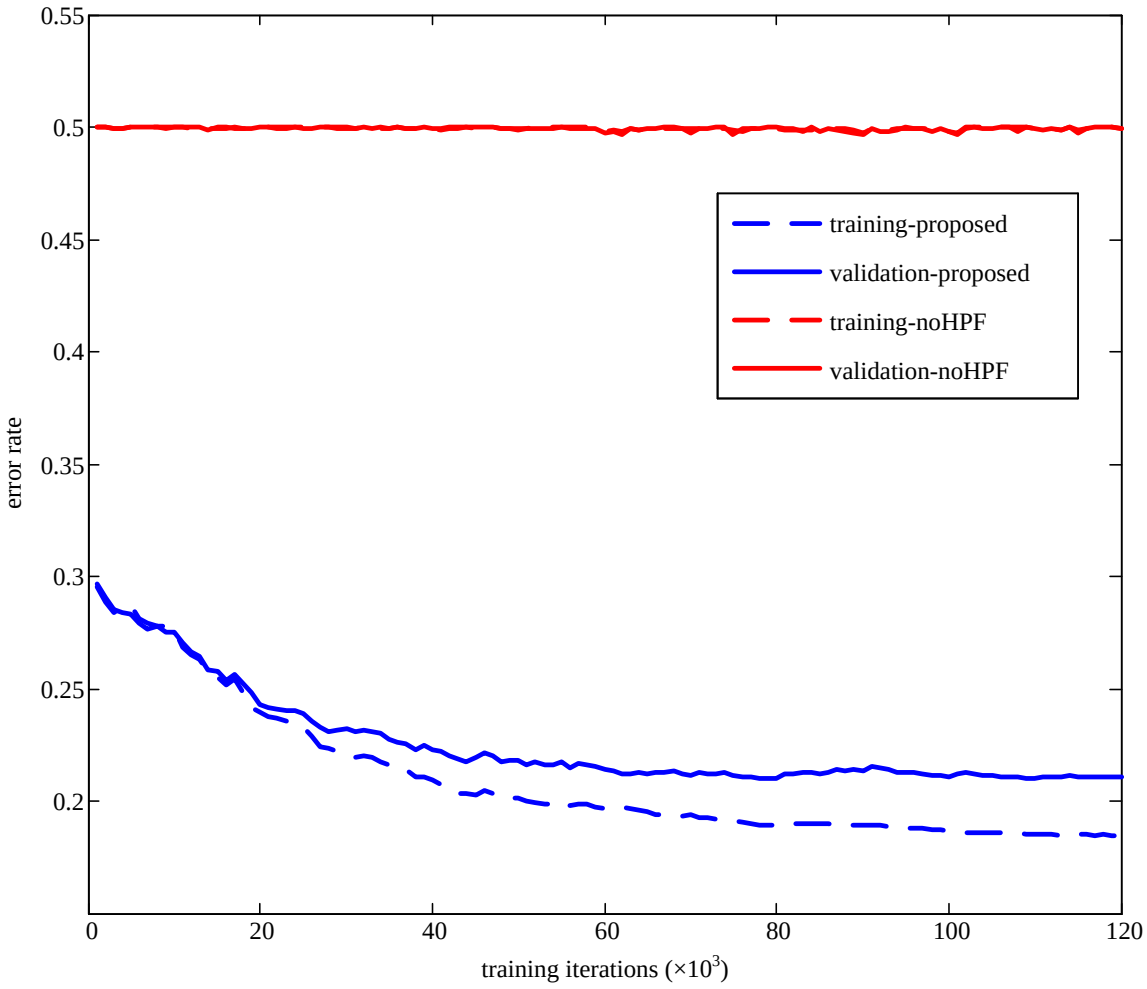


Figure 2.11 Training errors and validation errors: proposed CNN vs. the CNN without the HPF layer.

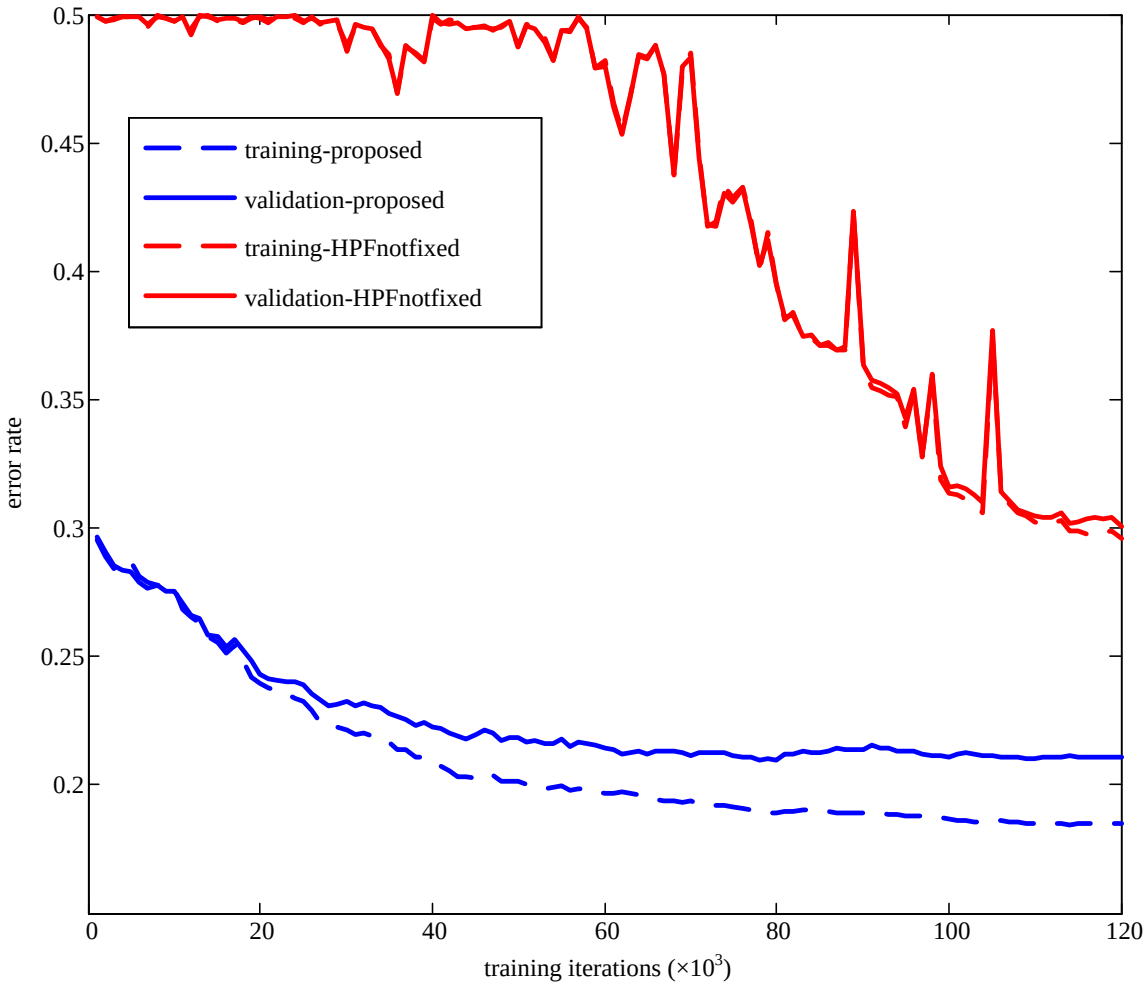


Figure 2.12 Training errors and validation errors: proposed CNN vs. the CNN without fixing the parameters in the HPF layer during training.

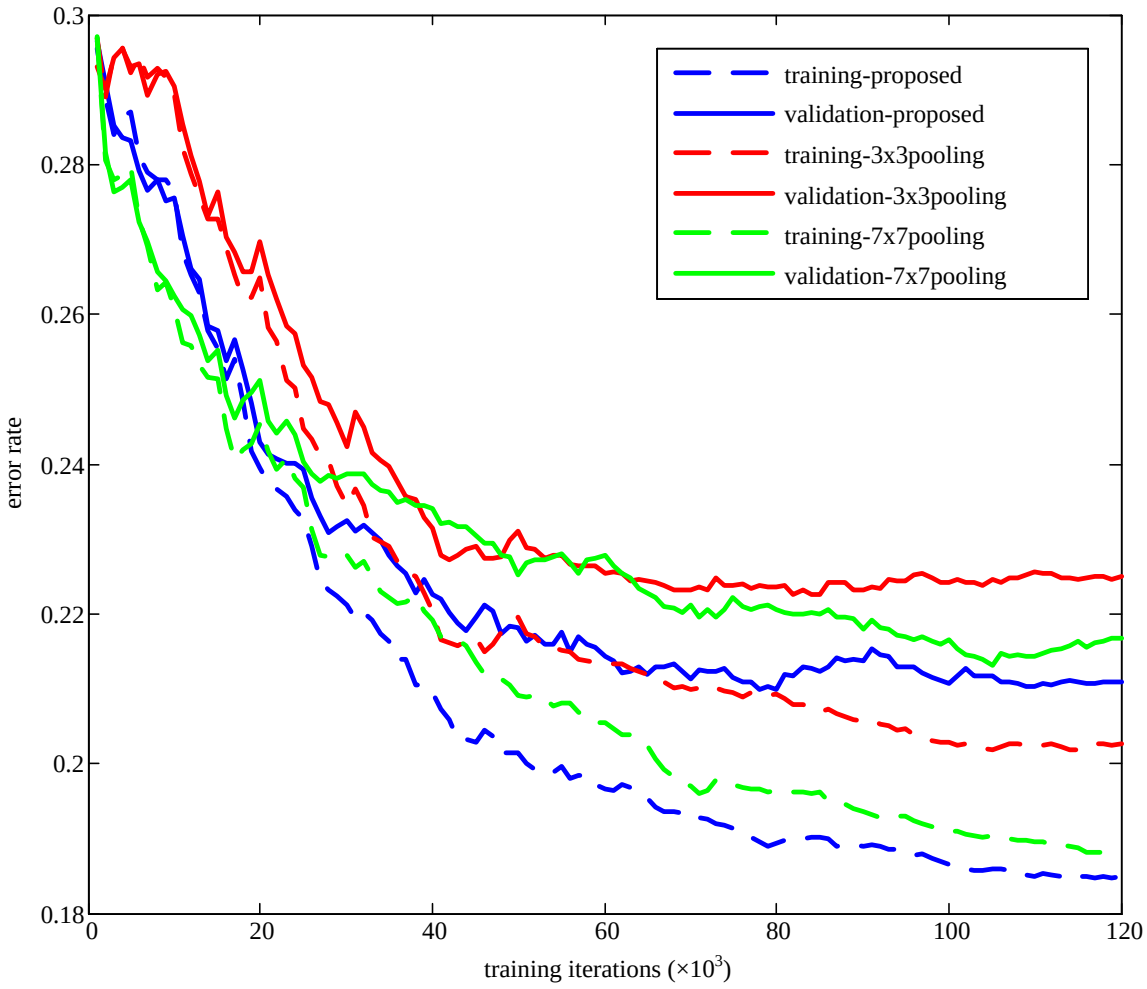


Figure 2.13 Training errors and validation errors: proposed CNN vs. the CNN with pooling sizes of 3×3 and 7×7.

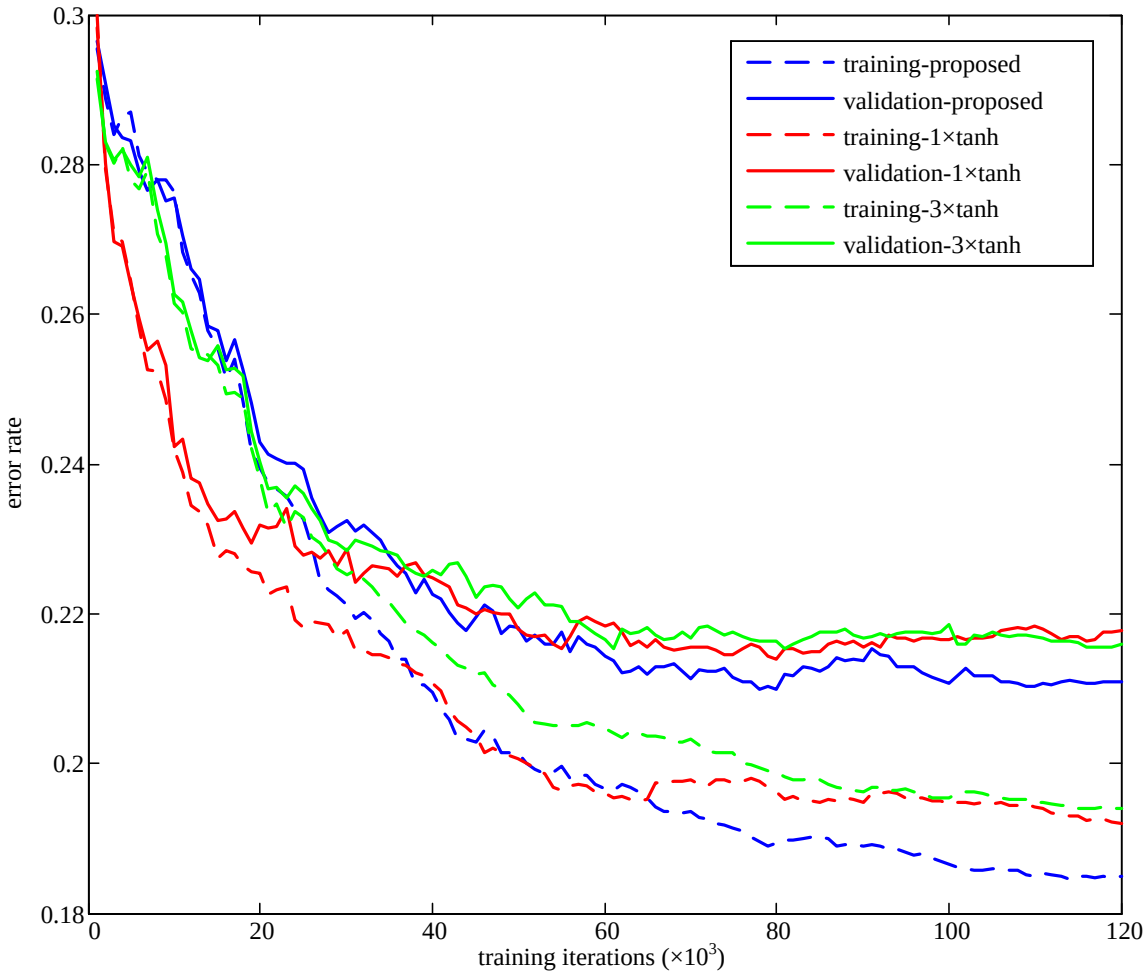


Figure 2.14 Training errors and validation errors: proposed CNN vs. the CNN with *TanH* only in Group-1 and the CNN with *TanH* in Group-1 – Group-3.

2.3.5 Dataset and Experimental Methods

We performed experiments using our designed CNN to detect two spatial domain content-adaptive steganographic algorithms: S-UNIWARD [3] and HILL [34], with embedding rates of 0.1 bpp and 0.4 bpp. The corresponding performance achieved by the SRM is used as reference. All of the experiments using the CNN reported here were performed on a modified version of Caffe toolbox [33].

The dataset used is the BOSSbase v1.01 [29] containing 10000 cover images of size 512×512. This dataset is the most widely used for the steganalysis and steganography research. The cover images are initially taken by seven cameras in the RAW format, and transformed to 8-bit grayscale images, then cropped to obtain the size of 512×512. Image data of the other class (stego) were generated through data embedding into the cover images. Hence, for each steganographic method and embedding rate, the dataset contains 10000 pairs of images.

Out of the 10000 pairs of images, 5000 pairs were set aside for testing to verify the performance (refer to Equation 2.7). These 5000 testing pairs were not touched in the whole training phase. The architectural design of the CNNs and selection of the components were done by performing a five-fold cross-validation on the training set. Once the optimal CNN architecture is determined, the corresponding five optimized CNN models, each of them trained on 4000 pairs in the training set and validated on the other 1000 pairs, form ensembles to classify the 5000 testing data. During the training process, the performance on the validation set was monitored from time to time, and the model that corresponds to the lowest validation error was saved and used for testing (the polyak averaging [119] was used to create the model for testing). More specifically, in the testing

stage, the 10000 testing images (5000 testing pairs) went through all the five trained CNNs one by one, and the output class-posterior probabilities were averaged for each test image to make the final prediction.

Mini-batch gradient descent was used to train all the CNNs in experiments. The momentum was fixed to 0.9. The learning rate was initialized to 0.001, and scheduled to decrease 10% for every 5,000 iterations, for all the parameters. A mini-batch of 64 images (32 cover/stego pairs) was input for each iteration. All of the CNNs were trained for 120,000 iterations. Parameters in convolution kernels were initialized by random numbers generated from zero-mean Gaussian distribution with standard deviation of 0.01; bias learning were disabled in convolutional layers and fullfilled in BN layers. Parameters in the last fully-connected (FC) layers were initialized using 'Xavier' initialization [37]. Except for the FC layer, weight decay (L2 regularization) was not enabled so that we could focus on designing the CNN architecture. For the same reason, no 'dropout' [106] was used.

2.3.6 Results

The experiments using the SRM (with ensemble classifiers [30]) were conducted on the same 5,000/5,000 train/test split as for the CNNs.

To evaluate the performance, we used the average accuracies recorded in Table 2.2, and the receiver operating characteristic (ROC) curves together with the corresponding area under ROC curves (AUC) illustrated in Figure 2.15. The ROC curves imply treatment of this classification problem as a signal detection problem, where cover images belong to the negative classes and the stegos belong to the positive classes. The average errors reported in Table 2.2 were obtained by comparing the averaged class-

posterior probabilities with the threshold equals 0.5; the ROC curves were empirically obtained through moving the threshold.

Overall, the CNN has better performance at relatively higher embedding rate compared with the SRM and ensemble classifier, and competitive performance at lower embedding rate. Table 2.3 reports the means and standard deviations of the testing accuracy obtained from the five single CNN models. Note that the results reported in Table 2.2 are ensemble results, whereas what is reported in Table 2.3 is the individual CNN results.

Table 2.2 Accuracies (in %) of CNN and SRM against S-UNIWARD and HILL

	0.1 bpp		**0.4 bpp**	
	CNN	**SRM**	**CNN**	**SRM**
S-UNIWARD	57.33	59.25	80.24	79.53
HILL	58.44	56.44	79.24	75.47

Table 2.3 Means and STDs of Single CNN Model Accuracies (in %)

	SUNIWARD		**HILL**	
	0.1 bpp	**0.4 bpp**	**0.1 bpp**	**0.4 bpp**
MEAN	56.85	79.03	57.55	77.58
STD	0.91	0.67	0.68	0.30

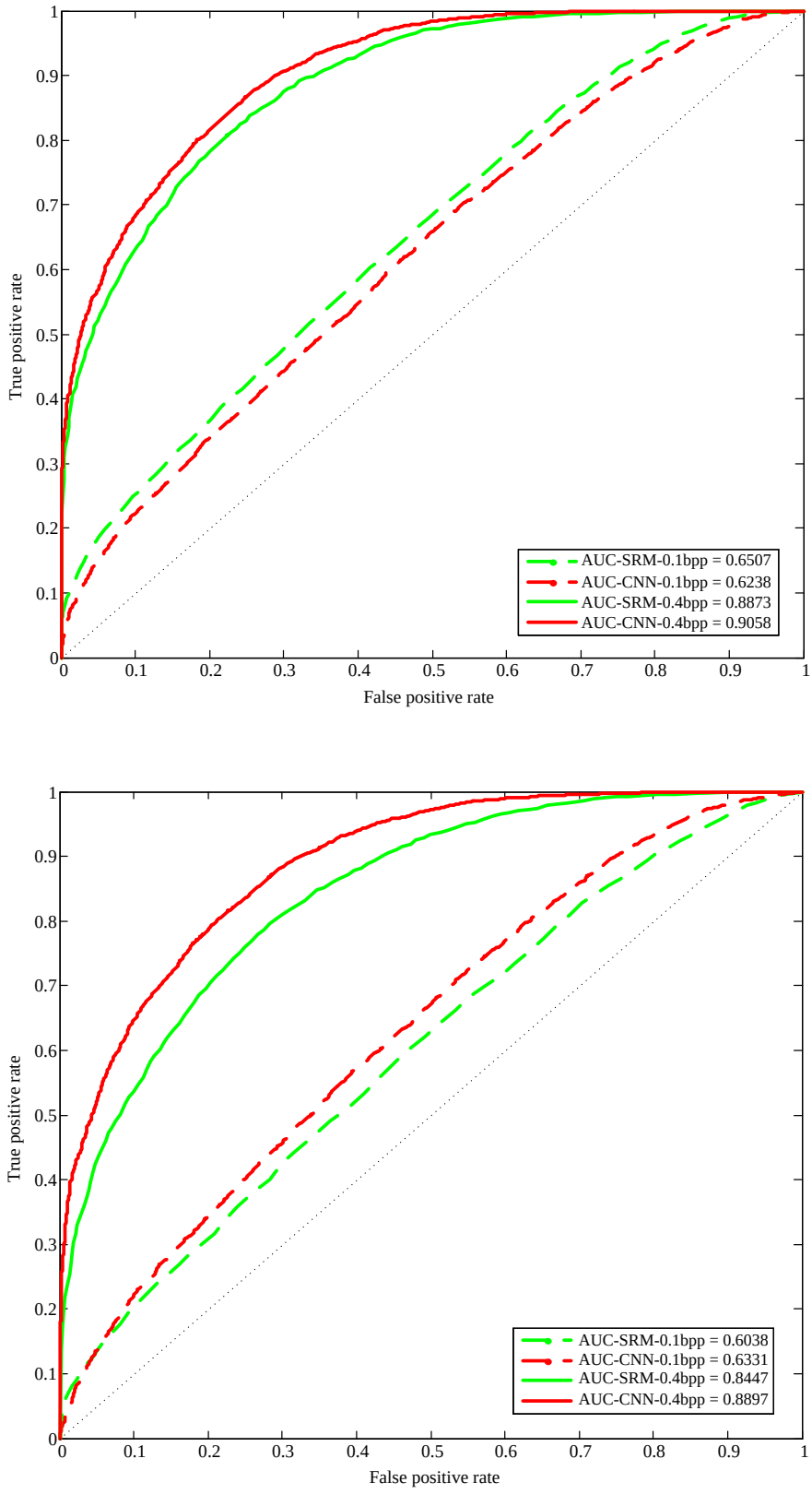


Figure 2.15 ROC curves. (Up) against S-UNIWARD. (Down) against HILL.

2.3.7 Conclusion

In this work, it has been shown that a well-designed CNN is a good steganalytic tool, and would have the potential, in the future, to provide a better detection performance. Currently, the proposed CNN is not fed with the probability maps of embedding derived in a similar manner as steganography like methods in [111–114]. Nevertheless, it would not be difficult to achieve this in the middle of the CNN architecture to further enhance the performance against content-adaptive steganography. For example, the embedding probability maps could be used in the pooling layers to perform weighted (by probabilities) average pooling. How to apply the CNN in the best way to defeat steganography in the JPEG domain [103, 115] would be another important future work.

We would like to emphasize that by no means should the architecture proposed in this work be deemed as optimal, e.g., using *TanH* right after the first two convolutional layers may not be the best choice when the other parts of the network change. Due to the strong coherence between network components, the best architecture always needs to be adjusted, but the philosophy of the design, holds. As the architectural design of neural networks is flexible, it is expected that in the future research better structures would be designed to further boost the performance of the CNN for steganalysis.

One of the drawbacks of the current CNNs is the fixed HPF kernel for noise residual generation. Learning from residuals instead of original images is itself suboptimal, as the high-pass filter which is not jointly optimized during training have caused information loss. It would also be of value to study how to let the CNNs learn from original images.

2.4 Ensemble of CNNs for Steganalysis

When performance is highly concerned, ensemble learning has been arguably one of the most widely adopted techniques to improve machine learning performance since the invention of boosting [42, 43], bootstrap aggregation [41] and random forest [4]. It is well-known that the neural networks, including the convolutional neural networks (CNNs), which have achieved great success in the fields of computer vision [23, 24, 36, 45, 46], are suitable to serve as base learners and form ensembles. In computer vision, the most prominent research studies focus on designing efficient CNN architectures, and seeking ways to improve the optimization efficiency of deep neural networks [45, 46]. Nevertheless, ultimate performance is always brought by ensembles of multiple CNNs. For example, all the winning solutions in the ImageNet Large Scale Visual Recognition Challenge [24, 45, 46] from year 2012 to 2015, are ensembles of multiple CNNs.

Inspired and encouraged by the success of CNNs in computer vision, the forensics society have started devoting research efforts on migrating the CNNs to solve forensics and steganalysis problems [26, 31, 47]. In [47], the proposed CNN boosted accuracy on detecting median filtering processing in images by 1% – 8% compared with previous works. In [26] and [31], attempts were made in applying CNNs to image steganalysis, although the reported performances are still worse than the traditional feature-based methods [12, 14, 48, 49]. All of those works perform classification using only a single CNN for each individual experiment. In [50], the architectural design of CNNs for steganalysis was discussed, and the ensemble (five CNNs) performance of the proposed CNN is competitive compared with that achieved by the SRM and FLD-ensemble [12].

The ensemble method used in [50] is the simple model averaging (averaging the output class-posterior probabilities of each CNN).

In this work, we go beyond model averaging, and test the performance of second-level classifiers trained on the feature vectors generated from base learners (CNNs). The feature vectors come from

1) the output posterior probabilities of the trained CNNs;
2) the output posterior probabilities of the CNNs with offsets in the spatial subsampling step of pooling layers;
3) the output vectors of the convolutional modules in CNNs.

The second one aims at recovering the information loss caused by spatial subsampling.

The performance of all the proposed ensemble methods is evaluated on BOSSbase [28] by detecting S-UNIWARD [3] at 0.4 bpp embedding rate. Results have indicated that both the recovery of the information loss caused by subsampling, and learning from features representations within CNNs instead of output probabilities, have led to performance improvement. While only tested on one dataset with a special steganalysis problem, the proposed ensemble methods should be generically applicable to most of the image steganalytic and forensic tasks using CNNs.

The rest of this chapter is organized as follows. In Section 2.4.1, we briefly review the CNN architecture used to build base learners, more details of the CNN design can be found in Section 2.3.1. All the ensemble methods we have studied are listed in Section 2.4.2. Dataset and settings for experiments are given in Section 2.4.3. Experimental results and discussions are presented in Section 2.4.4. Conclusions are drawn in Section 2.4.5.

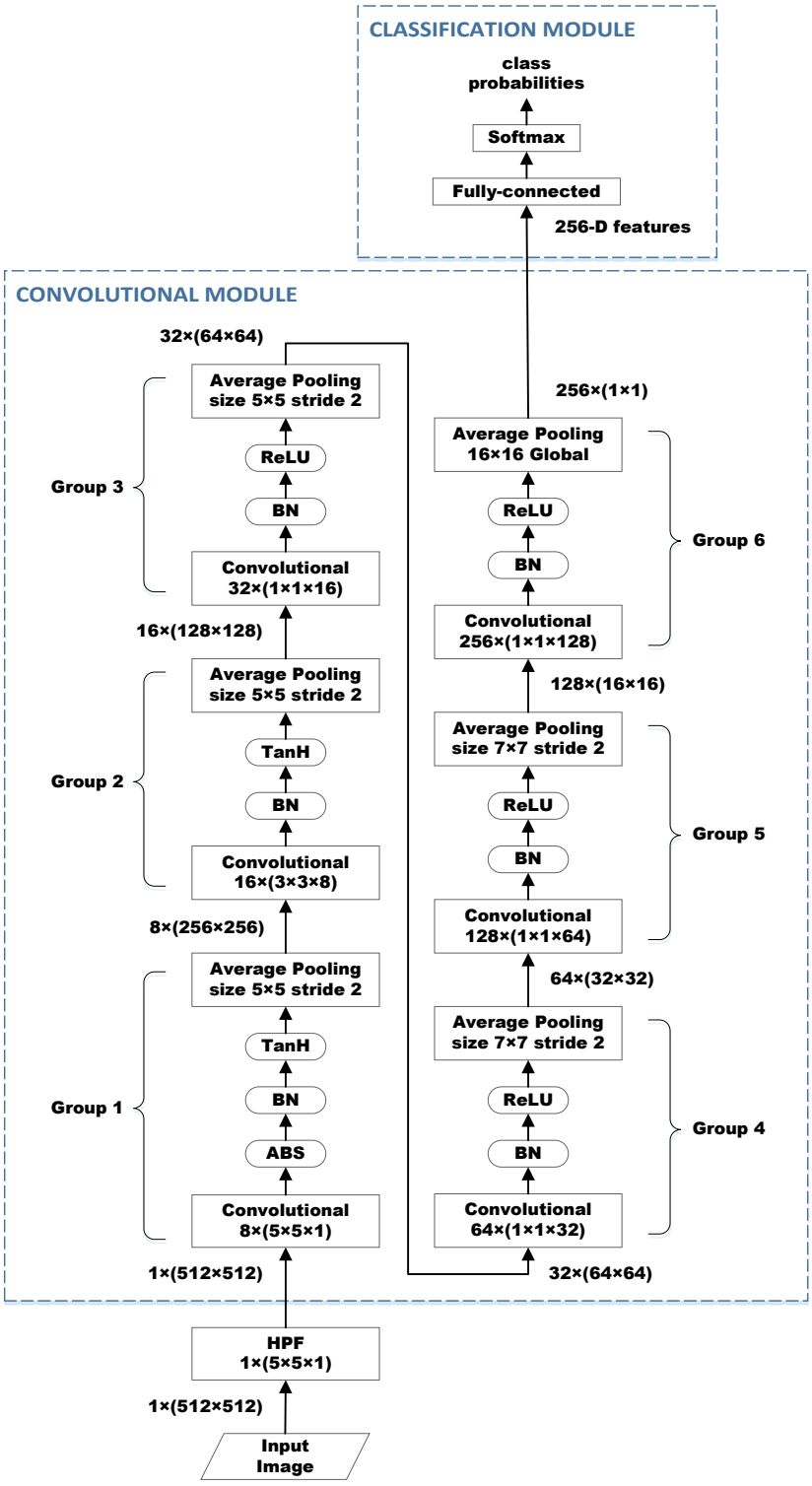


Figure 2.16 The CNN architecture for training of base learners. Inside boxes are the layer functions. Data sizes are displayed on the two sides. Sizes of convolution kernels in the boxes follow (number of kernels) × (height × width × number of input feature maps). Sizes of data follow (number of feature maps) × (height × width).

2.4.1 The CNN as Base Learner

The CNN architecture used in this work is almost same as that proposed in Section 2.3.1 except that we append one more group of layers (Group-6 in Figure 2.16) to the end of the convolutional module, and increase the pooling sizes of the last two pooling layers from 5×5 to 7×7. This work aims at studying strategies for ensemble learning instead of designing CNNs. To make this work self-contained, we briefly review the CNN architecture used for generating base learners. More details have been presented in Section 2.3.1.

The overall architecture is illustrated in Figure 2.16. A high-pass filtering (HPF) layer using the previously developed high-pass kernel [12] is placed at the very beginning to transform original images to noise residuals. The parameters in the HPF layer are not optimized during training; this CNN actually learns from the generated noise residuals instead of from the original images. Hence, in the rest of this work, the training data refers to the obtained residuals from the original images. The whole CNN contains a convolutional module responsible to transform the images/residuals to 256-dimensional (256-D) feature vectors, which serves as input to the linear classification module that generates a posterior probability output for each of the two classes given an image datum. Note that for binary classification problem, only one of the probability values is needed. The convolutional module comprises six groups of layers ("Group-1 – Group-6" in Figure 2.16), each of them starts with a convolutional layer, which doubles the number of spatial maps (often be referred to as the 'width' of a layer in CNN), and ends with an average pooling layer which performs local averaging as well as subsampling on the spatial maps (except Group-6). The CNN is equipped with the hyperbolic tangent (*TanH*)

as non-linear activations for Group 1 and Group 2, and the rectified linear unit (*ReLU*) activations for Group 3 – Group 6. An absolute activation (ABS) layer is inserted in Group 1 to force the CNN to take into account the (sign) symmetry existed in noise residuals. Immediately before each non-linear activation layer, the feature maps are normalized with batch-normalization (BN) [36].

Through global averaging, the pooling layer in Group 6 merges each spatial map to a single element: 256 maps of size 16×16 to 256-D features. In this work, we represent the size of the CNN by the output size of the last pooling layer (in Group 6), hence, we call the CNN in Figure 2.16 as 'SIZE 256'; 'SIZE 128' refers to a CNN with only half the widths for each layer and has roughly one quarter of the total number of parameters existed in 'SIZE 256'.

2.4.2 Ensemble Methods

In this section, we discuss in detail all the ensemble methods we have studied in this chapter.

Let $\{\mathbf{x}_i, y_i\}_{i=1}^{N}$ denote the training dataset, where *N* is the total number of training data which are the residuals generated by the HPF layer, $\mathbf{x}_i$ represents the *i*-th residual of the training set, and y_i is the corresponding binary label. Note that, for $i \in \{1,...,N\}$, we have $\mathbf{x}_i \in \mathbb{R}^{H\times W}$ and $y_i \in \{0,1\}$. The total number of CNNs trained and used as base learners is *T*, in this work, we choose $T = 16$. Denote the *k*-th CNN as h_k, which maps each residual image to a probability value, and is represented by $h_k = h(\mathbf{x};\mathbf{w}_k): \mathbb{R}^{H\times W} \to p \in [0,1]$, in which the parameters $\mathbf{w}_k$ is optimized by minimizing

the log loss denoted generally by $L_{\mathcal{H}} = L(h, y)$. The procedures to generate base learners (CNNs) are summarized below:

---

1. **for** k = 1 to T **do**
2. Generate a random permutation $\{\pi(i)\}_{i=1}^{N}$ of $\{1,...,N\}$.
3. Train h_k specified by its parameters $\mathbf{w}_k : \mathbf{w}_k = \arg\min_{\mathbf{w}} L_{\mathcal{H}}\left(h\left(\mathbf{x}_{\pi(i)}; \mathbf{w}\right), y_{\pi(i)}\right)$, where i = 1,…,N^{tr}, and N^{tr} is the size of the training set such that $N^{tr} < N$.
4. **end for**
5. Collect all trained CNNs $\{h_k\}_{k=1}^{T}$ as base learners.

Note that this process is almost same as bootstrap aggregation [41], in which each base learner is trained on a subset randomly drawn by sampling with replacement from the original training set. In this work, sampling without replacement is used instead.

Once the training of CNNs is completed, the most straightforward and commonly used ensemble strategy is to average the output probabilities from each CNN and compare the result with th = 0.5 to determine the corresponding class label which is equivalent to choosing the larger class posterior, i.e., for each test data $\mathbf{x}_t$, its label can be estimated by

$$y_t = \begin{cases} 0 & \frac{1}{T}\sum_{k=1}^{T} h_k(\mathbf{x}_t) < th \\ 1 & \frac{1}{T}\sum_{k=1}^{T} h_k(\mathbf{x}_t) \geq th \end{cases} \tag{2.20}$$

This is the basic ensemble method performed in our experiments. Note that this basic model averaging strategy does not require further learning. Next, we will show that besides simple model averaging, more can be dug out from the CNNs for steganalysis tasks.

CNNs usually adopt several subsampling steps to reduce the spatial dimensions and facilitate classification. These subsampling steps are fulfilled in pooling layers or convolutional layers with strides (subsampling rate) set larger than 1. In computer vision and other related research areas, the subsampling steps may not have negative effect, because they discard irrelevant information and help the optimization in deeper layers focus. However, as steganalysis relies on statistics, spatial subsampling could cause information loss, even after the information of skipped pixels have been encoded into neighboring pixels through, e.g., averaging. The dilemma is, it seems that the spatial subsampling is unavoidable, because without it, the statistical modeling in CNN would grasp the location information of embedded pixels from the training data. To help alleviate this issue, one possible solution is, given a trained CNN, we generate probability output with every possible subsampling offset so that every skipped pixel location could be covered once, as illustrated in Figure 2.17 (b – e). To make the explanation easy to follow, we assume the pooling regions to be 2×2 and the stride equals 2 for both horizontal and vertical direction. During training [Figure 2.17 (a)], the pooling layer sticks to only one set of spatial subsamples, i.e., a_{11}, $a_{13,}$ $a_{31,}$ a_{33}… For average pooling, the output of this pooling layer is calculated as $b_{11} = (a_{11} + a_{12} + a_{21} + a_{22}) / 4$. Because of the fixed offsets, the skipped locations, e.g., $b_{12} = (a_{12} + a_{13} + a_{22} + a_{23}) / 4$, is never used in training. The solution is to output probability values of all the four constellations of

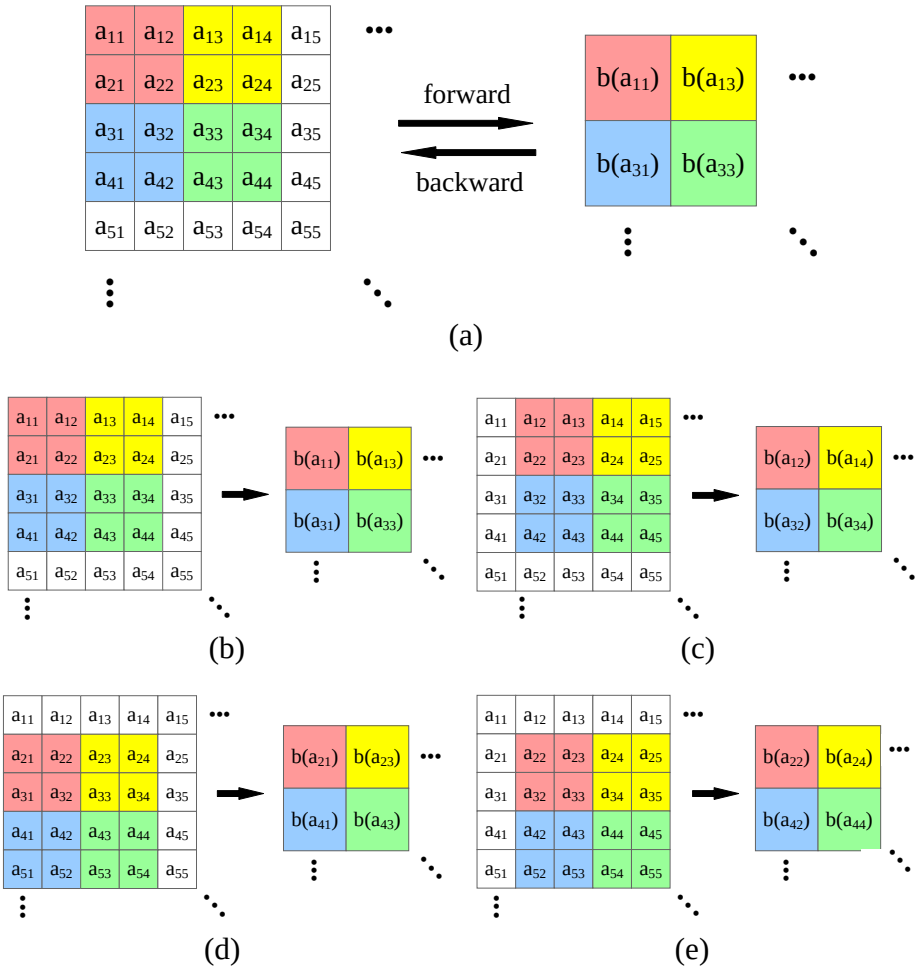


Figure 2.17 Pooling with local size 2×2 and stride 2. (a) Forward and backward passes of a pooling layer with fixed sampling locations in the training stage of the CNNs. (b) – (e) The four possible sampling when transforming image data with CNNs into feature vectors for ensemble learning.

subsampling for ensembles, as demonstrated in Figure 2.17 (b–e). Following this, given P pooling layers in a CNN, and assume stride equals 2, the total number of output (probabilities), M, generated from each trained CNN equals 4^P, for the CNN illustrated in Figure 2.17, we have P = 5 (in Group-1 – Group-5), and therefore M = 1024. In this scenario, using the averaging strategy, the class label is estimated as:

$$y_t = \begin{cases} 0 & \frac{1}{TM}\sum_{k=1}^{T}\sum_{d=1}^{M} h_k^d\left(\mathbf{x}_t\right) < th \\ 1 & \frac{1}{TM}\sum_{k=1}^{T}\sum_{d=1}^{M} h_k^d\left(\mathbf{x}_t\right) \geq th \end{cases} \tag{2.21}$$

One might realize that only 1 out of *M* cases is fully optimized during training [Figure 2.17 (a)] for each CNN, the others are close to the optimized because of the strong spatial correlations but are still suboptimal. In this case, it would be beneficial to map the original training data by the CNNs into a new feature representation and train second-level classifiers for optimal performance, as summarized:

---

1. Map the original training data $\left\{\mathbf{x}_i\right\}_{i=1}^{N}$ with base learners into $\left\{\mathbf{z}_i\right\}_{i=1}^{N}$, where $\mathbf{z}_i = \left\{h_k^1\left(\mathbf{x}_i\right), h_k^2\left(\mathbf{x}_i\right), ..., h_k^M\left(\mathbf{x}_i\right)\right\}_{k=1}^{T}$, for $i \in \{1, ..., N\}$.
2. Build a classifier using $\left\{\mathbf{z}_i, y_i\right\}_{i=1}^{N^{tr}}$.

In this work, the ensemble classifiers using fisher linear discriminant as base learners (FLD-ensemble) [17] developed specifically for steganalysis is used as the second-level classifier because of its good performance and efficiency. We have also tested linear support vector machines whose performance is roughly on par with the FLD ensemble.

The last ensemble strategy we are to test is to gather from each CNN the output of the last pooling layer, which is also the output of the convolutional module and input of the classification module as displayed in Figure 2.16. The intuition is that the FLD-ensemble are stronger compared with the linear classification module in the CNN.

Therefore, concatenating intermediate representations from every base learner CNNs before performing classification potentially increases the chance of mining more discriminative patterns. Let $f_k = f(\mathbf{x};\mathbf{w}_k): \mathbb{R}^{H\times W} \to \mathbb{R}^Q$ denote the function of the convolutional module in the *k*-th CNN, *Q* is the output dimension of the convolutional modules, this ensemble method can be summarized as:

---

1. Map the original training data $\{\mathbf{x}_i\}_{i=1}^N$ with base learners into $\{\mathbf{z}_i\}_{i=1}^N$, where $\mathbf{z}_i = \{f_k(\mathbf{x}_i)\}_{k=1}^T$, for $i \in \{1,...,N\}$.
2. Build a classifier using $\{\mathbf{z}_i, y_i\}_{i=1}^{N^{tr}}$.

2.4.3 Dataset and Settings

Training of the CNNs was performed on a modified version of Caffe toolbox [33]. Performance of the ensemble methods was evaluated by detecting S-UNIWARD [3] at 0.4 bpp embedding rate only, due to the long training time of CNNs and the long feature mapping time. It took about three weeks to run all the experiments using two NVIDIA Geforce GTX 980Ti graphics cards. The dataset used was BOSSbase v1.01 [28] containing 10,000 cover images of size 512×512. Image data of the other class (stego) were generated through data embedding into the cover images. Hence, the dataset contains 10,000 pairs of images. Out of the 10,000 pairs of images, 5,000 pairs were set aside for testing to verify the performance; the rest 5,000 pairs were used as the training set. To train each CNN as base learner, 4,000 out of the 5,000 training data were

randomly drawn, the rest 1,000 data were used as validation set to prevent the neural networks from overtraining. Two groups of CNNs with different network sizes were obtained: ‘SIZE 256’ and ‘SIZE 128’, the numbers refer to the output size of the convolutional module as explained in Section 2.4.1. A total of 16 CNNs were trained and used as base learners for both the two network sizes.

For reproducibility, information of the hyperparameters and settings used during training is summarized here. The learning rate was initialized to 0.001, and scheduled to decrease 10% for every 5,000 iterations. The momentum was set to 0.9. A mini-batch of 64 images (32 cover/stego pairs) was input for each iteration. All of the CNNs were trained for 120,000 iterations (960 epochs). Weight decay was not enabled except for the FC layers.

2.4.4 Results

In the first experiment, we study how the number of CNNs (as base learners) used for ensemble affect the performance. For simplicity, the basic model averaging strategy was adopted. For every fixed number of CNNs used for ensemble, we tested all the combinations (out of 16), and recorded the box plot for both of the two networks sizes. From Figure 2.18, we can conclude that increasing the number of CNNs for ensemble reduces variance, and consistently reduces detection errors. Comparing the performance of the two networks with different sizes, we observe that ‘SIZE 256’ has both better acuuracies and lower variance, which indicates that the width of a CNN is very important for steganalysis.

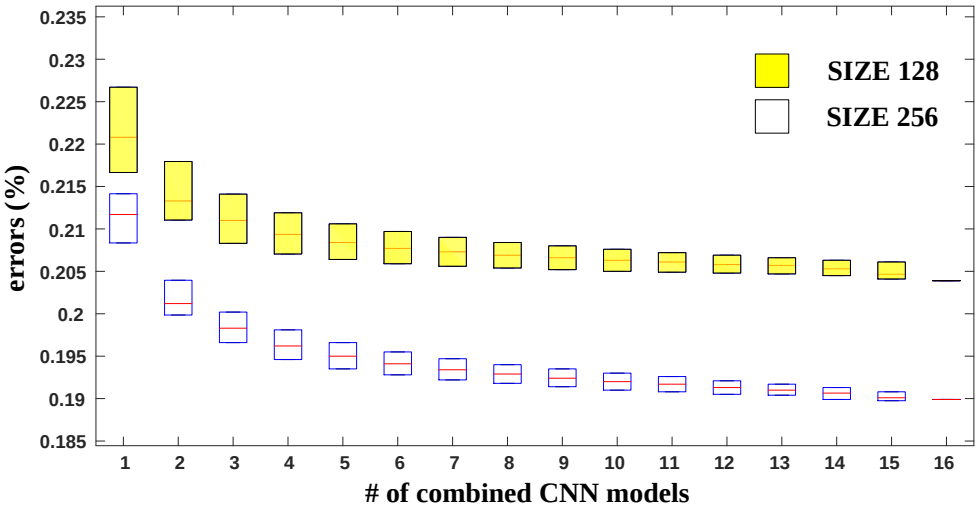


Figure 2.18 Box plots reflecting overall performance with different number of combined CNN models for both 'SIZE 128' and 'SIZE 256'. Red lines are the median values; the upper and lower bounds correspond to the 25 and 75 percentiles.

Table 2.4 Feature Dimensionality of Different Ensemble Scenarios

	SIZE 128		SIZE 256	
	Ensemble Methods		Ensemble Methods	
	AVE	**ENS**	**AVE**	**ENS**
PROB	16	16	16	16
PROB_POOL	16384	16384	16384	16384
FEA	N/A	2048	N/A	4096

Table 2.5 Error Rates of Different Ensemble Scenarios

	SIZE 128		SIZE 256	
	Ensemble Methods		Ensemble Methods	
	AVE	**ENS**	**AVE**	**ENS**
PROB	0.2039	0.1973	0.1899	0.1897
PROB_POOL	0.2018	0.1954	0.1918	0.1871
FEA	N/A	0.1906	N/A	0.1844

Table 2.4 records all the results using the ensemble strategies proposed in Section 2.4.2. In Table 2.4, the number of features for each ensemble scenario is presented. In

Tables 2.4 and 2.5, ***PROB*** refers to the direct CNN probability output. ***PROB_POOL*** refers to the subsampling method with offsets in the pooling layers for each CNN. ***FEA*** corresponds to the output features of the convolutional modules in CNNs. ***AVE*** means simple model averaging, and ***ENS*** is the FLD-ensemble [30]. From Table 2.5, we can summarize that the second-level learning consistently yielded better performance compared with model averaging. When the ensemble learning method was fixed to ***AVE***, ***PROB_POOL*** did not always have better performance over ***PROB***, probably due to the suboptimal probabilities output discussed in Section 2.4.2. The best performance was always achieved by learning from the concatenated features as output of the convolutional modules, which indicates that for performance, it might be preferred to abandon the linear classification modules in CNNs. To have some idea of where the presented ensemble performance are, the 34671-D SRM model [12] with the FLD-ensemble [30] on the same train/test split, has an error rate of 0.2047.

2.4.5 Conclusion

In this section, we study different ensemble strategies using CNNs as base learners for steganalysis. Results suggest that both the recovery of the lost information caused by spatial subsampling, and learning from intermediate feature representation in CNNs instead of output probabilities, improve the performance. While only tested on one dataset with a special steganalysis, the proposed ensemble methods should be generic and could be performed in most of the image forensic tasks using CNNs.

2.5 Potentials

Through sophisticated architectural design, our proposed CNN tailored for image steganalysis has begun to take the lead when compared with the most popular feature-based methods; by forming ensemble, the CNNs as base learners further expended the lead. So far, no published CNN design has performance near ours for steganalysis, though the improvement over traditional feature-based methods is not as significant as that has been reported in the field of computer vision. The performance of our CNN in its current form would naturally be improved by simply increasing the number of convolutional kernels (the 'width' of CNN), as has been shown in Figure 2.18, but at the cost of more memory consumption. To make this potential clear, we show in Figure 2.19 the steganalysis performances with different 'width' of the proposed CNN. Another 1–2% lower error rates would be expected if the 'width' can be further doubled and probably even more with ensemble learning, when better hardware is available. Besides, the performance would boost in favor of CNN-based steganalysis compared with feature-based methods, as can be already seen in [31], when more training data is available.

The other interesting function of the CNN is that the binary classification problem for steganalysis we are solving now can be extended to classify data embedding on pixel level of a given image. In computer vision, this extension corresponds to using the pre-trained CNN on image classification to achieve image segmentation [46, 120–124], which is rather straightforward. For steganalysis, this would mean from reporting if a given image has been data-embedded to precisely locating the pixel changes during embedding, which is a function that the feature-based methods does not have.

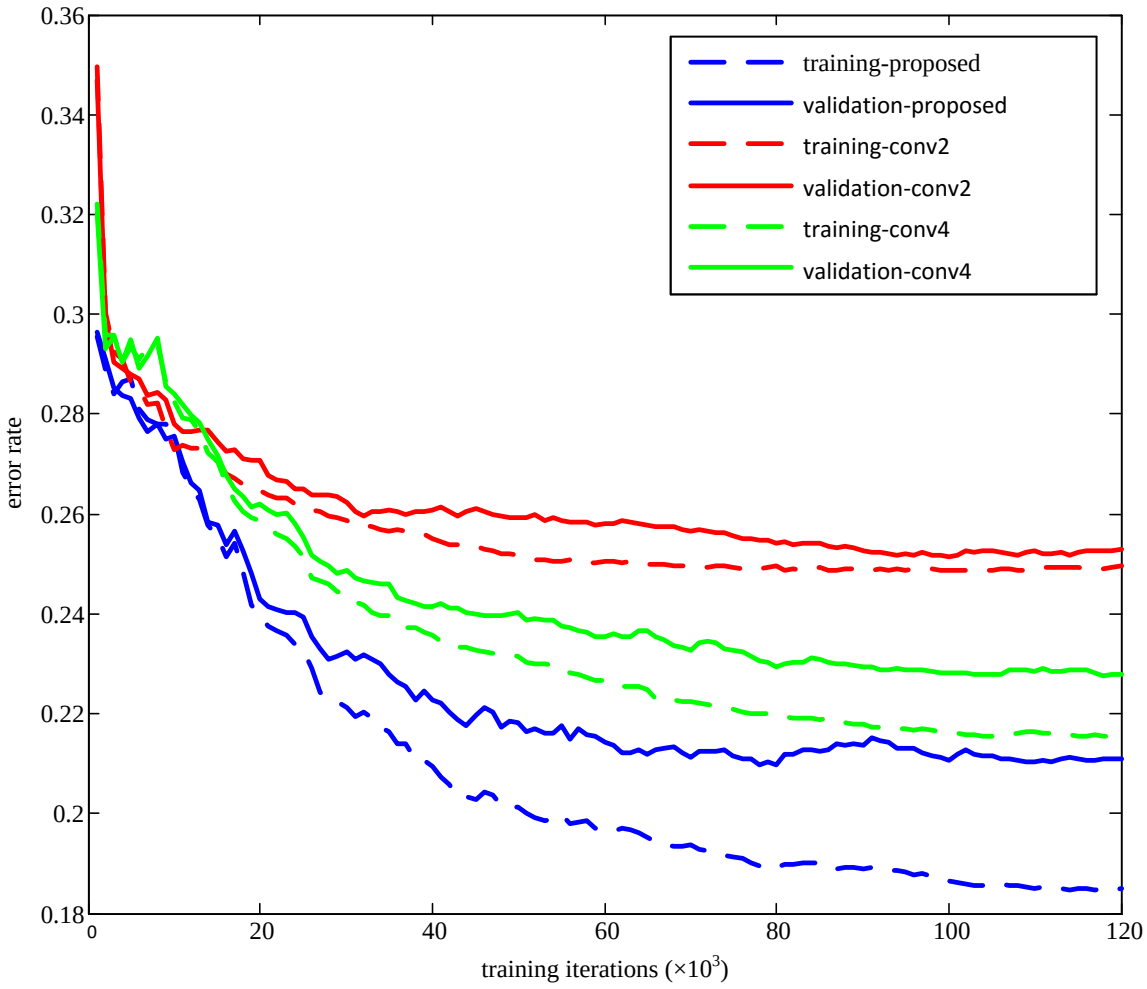


Figure 2.19 Training errors and validation errors: proposed CNN vs. the CNN with a half of the 'width' (conv4) and the CNN with a quarter of the 'width' (conv2).

CHAPTER 3

FEATURE-BASED CAMERA MODEL CLASSIFICATION

3.1 Introduction

Digital image producing devices such as cameras, cell-phones, camcorders and scanners are nowadays popular. As digital images are sometimes used as evidence in court, knowing the source and authenticity of the images used as evidence is important. However, the development of image editing software enables manipulation of both the contents and source information of digital images with ease, thereby compromising the credibility of them as evidence. Although embedding watermarks during image production to detect tampering is a possible solution, so far, they are not widely implemented by manufacturers of image producing devices. Hence, in most cases, we have to rely on blind and passive forensics on content of digital images for source identification and authentication.

In this chapter, we are to address the problem of source digital camera model classification, i.e., given an image, we need to figure out the source camera model that produced the image through a feature extraction and pattern recognition process that relies solely on the image content.

3.1.1 Literature Review

Figure 3.1 gives us an overview of a common imaging pipeline inside digital cameras. When light comes in, it first goes through a lens system that can cause straight lines to be rendered as curved lines in images. The fact that different lens system differs in this kind of geometrical distortion was used by Choi et al. [51] for camera model classification. In

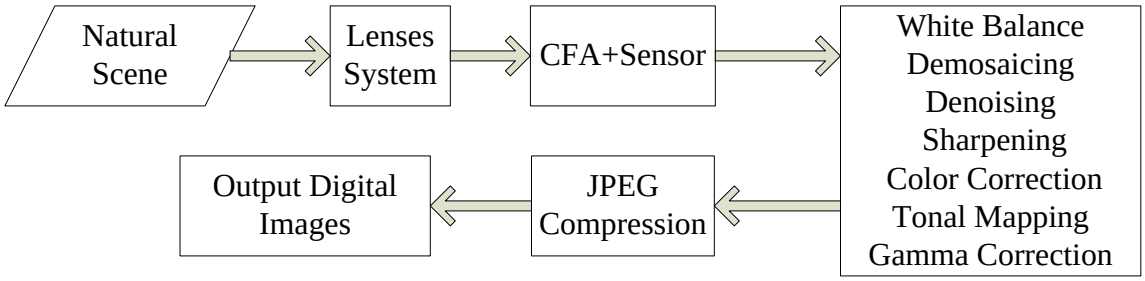


Figure 3.1 A common digital image producing pipeline.

[51], the three-camera classification accuracy reaches more than 91%. The drawback of this method is that detection accuracy depends highly on the existence and positions of straight lines in images.

After light comes out of the lens system, it goes through a filter system which consists of infra-red and anti-aliasing filters and possibly other types of filters. The output of the filter system is then input into a CCD or CMOS sensor by which it is transferred to electric signals. Filler et al. [52] considered the photo-response non-uniformity noise (PRNU) defined as different sensitivity of each pixel to the same light caused by the inhomogeneity of silicon wafers and imperfections during the sensor manufacturing process. In [52], seventeen different camera models from eight different brands were tested. The average classification rate is about 87%.

As sensors are of high cost, most digital cameras use only one sensor instead of three to record color images and a color filter array (CFA) that forms a checkerboard pattern is used in front of the sensor. By doing this, each pixel of an image only records one color component instead of three, and the other two color components can be recovered from nearby pixels by so-called demosaicing algorithms which are basically interpolations. Although the CFA usually adopts Bayer pattern [53], there is no standard

demoisaicing algorithms. Hence, camera manufactures design demosaicing algorithms for their own cameras. Inspired from the fact that different camera models adopt different CFAs and demosaicing algorithms, Swaminathan et al. [54], Long et al. [55], and Bayram et al. [56] make use of traces left by CFAs and interpolation (demosaicing) algorithms during image formation for camera model classification. In [54], linear interpolation coefficients are estimated through singular value decomposition and used as features for classification. Their algorithm can classify camera brands with an overall average accuracy of 90% for nine brands. Long et al. [55] compute autocorrelation of the modeling error by also assuming a linear interpolation model followed by a principle component analysis to find out the most important components of the coefficient matrices to serve as features. Five cameras from five different brands were tested, and the classification accuracy is over 95%. Bayram et al. [56] propose to estimate the color interpolation kernel using expectation–maximization algorithm, which was previously designed for image resampling (resizing) detection by Popescu et al. [57]. The average brand classification accuracy of three different cameras considered in [56] can reach 96% by assuming a 5x5 interpolation kernel. As most cameras output images in the JPEG format, besides color interpolation, the digital image processor also fulfils the task of JPEG image compression. Choi et al. [58] proposed to use the bit per pixel and the percentage of non-zero integers in each DCT coefficient as features for camera model identification. The average accuracy of classifying four camera models is about 92%.

Compared with the methods just mentioned, Kharrazi et al. [59] provided a more universal feature-based method which takes the whole image formation pipeline into consideration. From each image, 34 features including color features, image quality

metrics and wavelet coefficient statistics are extracted. The performance of a combination of the features proposed in [59] and six proposed camera white balancing features was thoroughly evaluated by Gloe et al. in [60] using a carefully designed dataset for benchmarking camera identification methods: the 'Dresden image database' [61]. In their experiments, 44 cameras spanning 11 camera models from the 'Dresden Image Database' were used. Based on carefully designed experiments, they draw the conclusion that this feature-based method does capture model information and is both practical and reliable. In [61], 96.42% average accuracy was reported using the same feature set with 18 camera models in the 'Dresden Image Database'.

In this dissertation, two advanced statistical feature-based camera model classification methods are presented in Chapter 3.2 and Chapter 3.3. Both of the two methods employ non-linear support vector machines (SVM) for classification, and effective statistical feature set are proposed as input for SVMs. The first statistical feature set is composed of Markov transition probabilities capturing the dependency between neighboring pixel values on the difference block DCT coefficients [62]. Elements of the transition probability matrices are directly used as features to build multi-class SVMs. The effectiveness of the proposed Markov feature set was verified by classifying eight camera models with a total of 40,000 images. The second feature set are composed of uniform gray-scale invariant local binary patterns [63] calculated from pixel values in both spatial and wavelet domains. Multi-class support vector machines were built for classifying eighteen camera models from the 'Dresden Image Database'. Classification performance showed that our proposed features outperformed feature set used in [61] and achieved state-of-the-art performance.

3.2 Markov Features in Block-DCT Domain

Since camera manufactures adopt different JPEG quantization matrices as well as different image processing algorithms within their camera models, which could result in statistical difference of the final JPEG quantized Block DCT coefficients, we propose a new set of statistical features capable of capturing the statistical difference of the quantized block DCT coefficients of JPEG images. Elements of Markov probability transition matrices are used here as the statistical features. Instead of directly calculating the probability transition matrices from the block DCT coefficients, we focus on the difference JPEG 2-D arrays which are actually the difference of the magnitude of the quantized block DCT coefficients. By taking difference, it is assumed that the statistical difference between camera models can be enlarged. For simplicity, in this work, only one-step Markov Process is considered and transition probabilities corresponding to large difference values are merged to prevent modeling less populated statistics as well as to achieve a great feature-size reduction. YC_bC_r is used as the color model in this work, where Y is the luminance component; C_b and C_r are the blue-difference and red-difference chrominance components. Probabilities in Markov probability transition matrix from four directions are extracted from the Y component and the C_b component of each JPEG image. Those features will then be used as the input of the classifiers.

The rest of this Section is organized as follows. In Section 3.2.1, details of Markov feature extraction together with the whole classification workflow are presented. Experimental results and some more empirical studies are reported in Section 3.2.2 and Section 3.2.3 respectively. Conclusions are drawn in Section 3.2.4.

Figure 3.2 The four directions considered for Markov transitions.

3.2.1 Markov Features

In this section, we first consider where to extract effective statistical features in order to capture the statistical difference for camera models classification purpose.

Instead of extracting statistical features directly from quantized block DCT coefficients, features are extracted from the difference JPEG 2-D array. JPEG 2-D array can be calculated by taking the absolute value of each quantized block DCT coefficient. Because the contents of all the images vary a lot and differ from each other, which are not desired for camera model classification, to reduce the influence of image content, we introduce the difference JPEG 2-D array, which is defined by finding the difference between an element and one of its neighbors in the JPEG 2-D array. All the four directions are considered, namely, horizontal, vertical, main diagonal and minor diagonal, as shown in Figure 3.2. Denote the JPEG 2-D array generated from a given test image by **X**, and the element of it by $X_{i,j}$, $i \in \{1,...,H\}$ and $j \in \{1,...,W\}$, where H and W are the height and width of the JPEG 2-D array. Difference arrays are generated from the four directions. For example, elements in the horizontal difference JPEG array $Y_{i,j}^{h}$ can be calculated as

$$Y_{i,j}^{h} = X_{i,j} - X_{i,j+1} \tag{3.1}$$

The other three difference arrays can be calculated in the same way.

It is expected that the image content influence can be reduced largely by considering the difference between an element and one of its neighbors in the JPEG 2-D array, in the meantime, the statistical difference caused by different camera pipelines is increased, resulting in better discrimination. The negative points of this operation are that it inevitably enhances the interference brought by high-frequency regions which heavily depend on individual image content, as well as increase camera noise that does not reflect camera model information, neither might be ideal to characterize camera models; fortunately, results show that the positive part dominate in the performance, furthermore, we have also include a truncation step to limit values of large magnitude in the difference maps to weaken the negative points.

We would like to emphasize that those four difference arrays are not calculated directly from the quantization block DCT coefficients, but from the JPEG 2-D arrays, which consists of the magnitudes of quantized block DCT coefficients. There are three reasons that we take absolute values (element-wisely) before calculating the difference:

- The magnitudes of the DCT coefficients decrease along the zig-zag scanning; this characteristic can be more easily captured by taking absolute before calculating difference.
- Taking absolute value before calculating difference can to some extent reduce the dynamic range of the output 2-D arrays compared with the 2-D arrays generated by calculating difference from the original block DCT coefficients directly.
- The signs of DCT coefficients mainly carry information of the outlines and edges of the original spatial domain image. Note that the outlines and edges are related only with the contents of images, they carry little useful information for camera model classification.

Hence, by taking absolute values, we keep the information regarding camera models and suppress the influence of image contents.

Now we talk about how to extract effective features from difference JPEG 2-D arrays. It is known that the BDCT coefficients have been de-correlated. However, there still exists intra-block correlation [64] within a local block. Therefore, we propose to model the difference JPEG 2-D arrays using Markov process, which takes into consideration the correlations among the coefficients. Markov process can be specified by the transition probabilities. For simplicity, here we only consider one-step Markov process, i.e., only one direct neighbor for each element within difference JPEG 2-D arrays is considered. As there are four difference JPEG 2-D arrays calculated from four directions, the transition probability matrices are calculated from their corresponding difference JPEG 2-D. Thus, totally we can generate four transition probability matrices from each JPEG 2-D array. Those transition probabilities are used as features for classification.

The size of a transition probability matrix depends on the number of different values. In the difference JPEG 2-D array, the number of possible different values is very large, resulting in a huge amount of sparsely populated transition probabilities, which is not ideal for the following pattern recognition process because of the high dimensionality and less accurate (due to the sparsity of the probability statistics) features. Figure 3.3 shows the normalized average histograms of horizontal difference JPEG 2-D arrays on the *Y* components calculated from 40,000 images; this roughly tells us the distribution of the values within a difference JPEG 2-D array. Since the distribution is Laplacian-like,

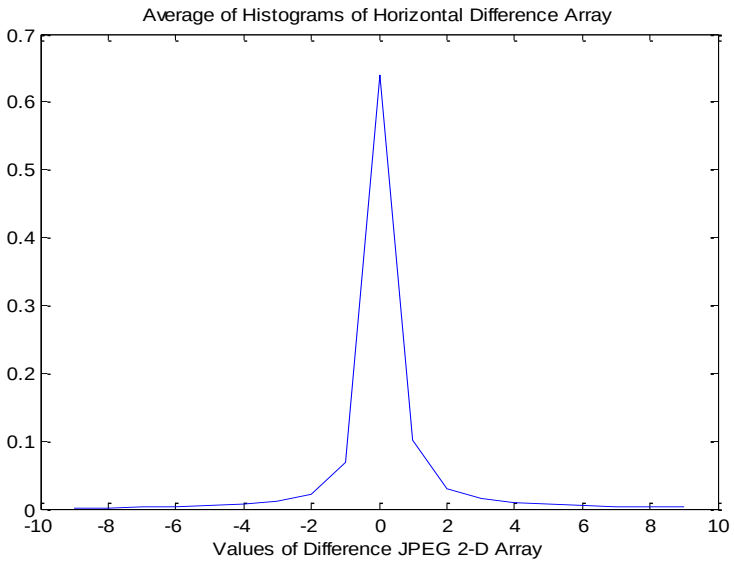


Figure 3.3 Distribution of horizontal difference arrays.

we merge the big values with truncation to limit the range of values from $-T$ to $+T$ with the following equation:

$$trunc(x) = \begin{cases} T & x \geq T \\ -T & x \leq -T \\ x & otherwise \end{cases} \tag{3.2}$$

Those values that are either smaller than $-T$ or large than $+T$ are forced to be $-T$ and $+T$, respectively, so as to keep as much information as possible. This truncation step achieves balance between complexity and performance, and results in a transition probability matrix of dimensionality $(2T+1)^2$. The conditional probabilities generated from a difference JPEG 2-D array in horizontal direction, e.g., the probability of the right neighbor $X^h_{s,t+1} = n$ when the current pixel $X^h_{s,t} = m$ are calculated by

$$\Pr\left\{X^h_{s,t+1}=n \mid X^h_{s,t}=m\right\}=\frac{\sum_{i,j} I\left(X^h_{i,j+1}=n, X^h_{i,j}=m\right)}{\sum_{i,j} I\left(X^h_{i,j}=m\right)} \tag{3.3}$$

where $m,n \in \{-T,-T+1,\ldots,0,\ldots T-1,T\}$. Note that the directions of Markov modeling and difference JPEG arrays are kept same. Again, probability values for the other three directional difference JPEG 2-D array can be calculated in the same way.

When images are compressed inside cameras, the first step is to convert images from *RGB* color model to YC_bC_r model. Therefore, it is natural to extract features from *YCbCr* representation. The proposed feature set considers transition probability matrices of all the four directional difference JPEG 2-D arrays from *Y* component. There is also some useful information for classification in C_b and C_r color components. Since C_b and C_r color components are usually processed in the same way in cameras, features generated from C_b and C_r are heavily correlated. In our work, only C_b component is considered in the feature extraction process. Furthermore, since both of the two color components have been downsampled during compression, only horizontal and veritcal directions of difference JPEG arrays are considered for C_b component, resulting in further reduction of complexity. In summary, from *Y* component, four transition probability matrices are generated, each corresponds to one direction. Given that the truncation threshold are set to $T = 4$ (detailed study of selecting the proper threshold will be shown in the section of empirical studies), there are $(2T+1)\times(2T+1) = 81$ probability features in each of these four transition probability matrices. In total, we have $4\times(2T+1)\times(2T+1)=324$ probability features from *Y* component of an image. As we only

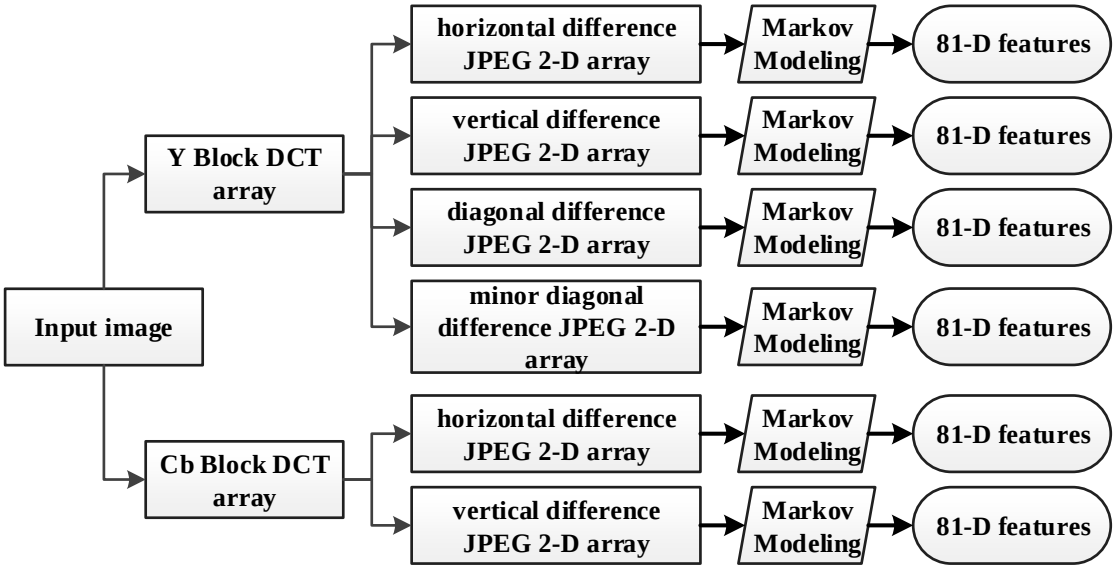


Figure 3.4 Block diagram for feature extraction.

consider two directions for Cb component, $2\times(2T+1)\times(2T+1)=162$ probability features are generated from C_b component. Combining all the features generated from Y and C_b components together, totally 324+162=486 probability features are generated from each image. The block diagram of the feature extraction process is given in Figure 3.4.

3.2.2 Results

Before large-scale experiments, a light-weight study has been carried on to show the discriminative ability of the proposed Markov features with image data taken by the author in controlled manner. Nikon Coolpix L18 and Nikon Coolpix S50 were selected as two camera models for study. Each camera took 75 images; all the images form pairs that recorded exactly the same scenes by the two cameras. This guarantees that classification would not be affected by different image content but focuses on characterizing camera models. Transition probability matrices of horizontal and vertical difference directions from Y component were considered. All the probabilities corresponding to the same data

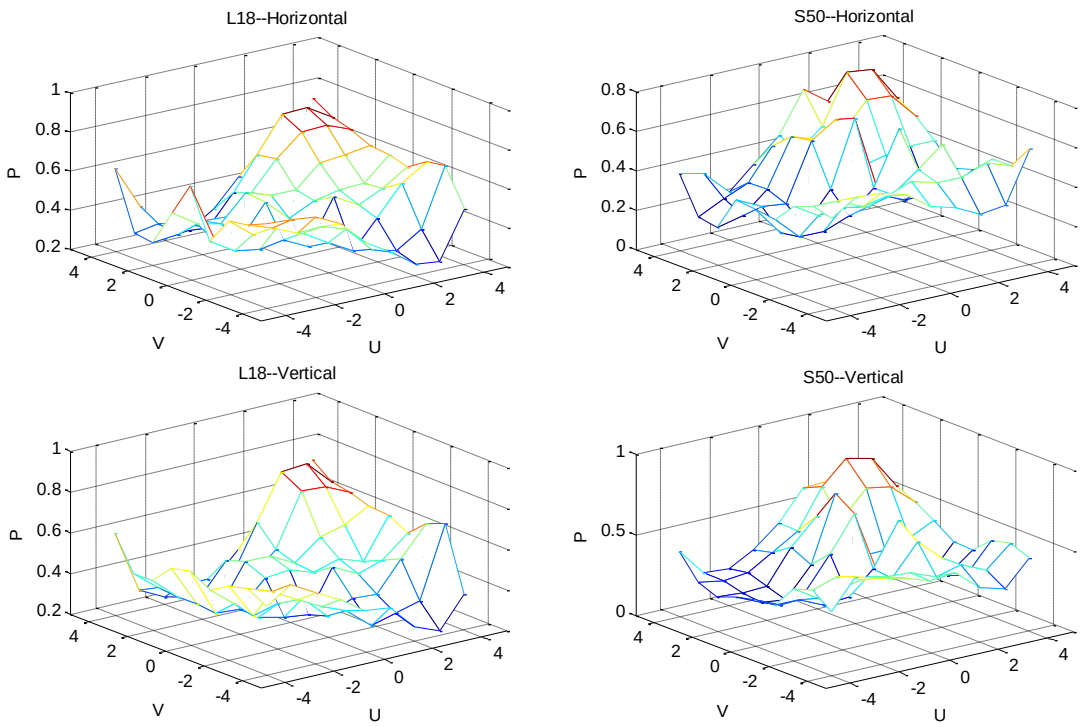


Figure 3.5 A visual comparison of the transition probabilities for the two camera models.

values were scaled and averaged together. Figure 3.5 gives us a visual comparison of the shapes of the transition probability matrices along horizontal and vertical directions of these two cameras. U and V axes are values in the difference JPEG arrays and P axis is the probability values. The difference of the shapes can be easily observed in both the horizontal and vertical directions, which proves the effectiveness of our proposed model. This kind of observation is an important motivation to large scale experiments.

For large-scale experiments, all the classification was accomplished by support vector machines (SVM) equipped with polynomial kernel, and the proposed Markov features serve as input to the SVMs.

Table 3.1 Confusion Matrix Using the Proposed Markov Features

	Kodak 6490	**Kodak Z740**	**Nikon D40**	**Nikon 3200**	**Nikon 4600**	**Sony P200**	**Canon 350D**	**Canon SD750**
Kodak 6490	**81.5**	17.2	*	*	*	*	*	*
Kodak Z740	14.4	**84.6**	*	*	*	*	*	*
Nikon D40	*	*	**95.7**	*	*	*	*	*
Nikon 3200	*	*	*	**93.7**	4.3	*	*	*
Nikon 4600	*	*	*	4.4	**93.1**	*	*	*
Sony P200	*	*	*	*	*	**98.4**	*	*
Canon 350D	*	*	*	*	*	*	**95.7**	*
Canon SD750	*	*	*	*	*	*	*	**97.5**

Table 3.2 Confusion Matrix for Camera Brand Classification Using The Proposed Markov Features

	Kodak	**Nikon**	**Sony**	**Canon**
Kodak	**98.9**	*	*	*
Nikon	*	**98.2**	*	*
Sony	*	*	**98.4**	*
Canon	*	*	*	**97.0**

In the literature, such as in [51, 55, 56, 58], each camera model was represented by only one camera. This is not ideal for the 'model' identification in practice, because the images produced by only one signal camera of each model might contain information of the individual camera besides the model information, therefore, the trained classifiers might not be able to correctly classify images produced by different cameras of the same models. In this work, the dataset were prepared in a more practical and rational way. We collected 5,000 images from each camera model. For each model, 5,000 images from 30 to 40 different cameras were used for experiments. Through this careful data collection,

we eliminated the chance of capturing the characteristics of a specific camera rather than the characteristics of a camera model. For each camera model, 4,000 out of 5,000 images were used for training classifiers, and the rest 1,000 for testing. There were totally eight different camera models from four manufacturers in the dataset, hence, totally there were 40,000 images. All these images were downloaded from www.flickr.com.

The classification results are given in Table 3.1 in the form of a confusion matrix. Each row in the confusion matrix corresponds to the actual camera models and each column corresponds to the predicted camera models. Percentages in the diagonal line marked in bold are the correct classification rate for each camera model. To make the form concise, we omit all the percentages smaller than 2%, this applies to all the confusion matrices in this section. By taking average along the diagonal lines, the average model classification accuracy is 92.5%. It can also be observed that most of the wrongly classified are within same camera brand (maker). This is reasonable because camera models with the same makers generally have similar hardware and image processing pipelines. Table 3.2 captures the confusion matrix for camera brand classification, the average brand classification accuracy reaches over 98%.

3.2.3 More Empirical Studies

In the proposed feature extraction approach, large values in difference JPEG 2-D were bounded and merged when calculating Markov probabilities, thereby avoiding modeling sparsely populated probabilities and reducing dimensions. To perform the truncation, a decent threshold is necessary to achieve the balance of performance and information loss. In this section, experiments have been conducted on how different threshold values affect the average model classification results as well as the information loss (proportions of the

Table 3.3 Relationship between Feature Dimensions, Average Classification Accuracies and Information Loss

	Feature Dimension	**Average Accuracy**	**Information Loss**
T=1	9	49.1%	19.1%
T=2	25	72.1%	14.1%
T=3	49	77.8%	11.4%
T=4	**81**	**80.3%**	**9.2%**
T=5	121	81.4%	8.7%

values in the difference JPEG 2-D array that fall out of the thresholding range). For simplicity, only features from horizontal difference JPEG 2-D array of Y component are considered, and feature size is $(2T+1)^2$. In Table 3.3, relationship between feature dimensions, average classification accuracies and information loss is shown. Note that dimensions of feature vectors grow quadratically with the increase of the threshold value. Comparing the cases $T = 4$ and $T = 5$, the corresponding dimensions differ by 40, while the classification accuracies differed by less than 1% and only 0.5% more values of coefficients fell out of the threshold range. Therefore, $T = 4$ is a proper choice.

The next study is about the correlation between Markov probabilities extracted from two color components. In Section 3.2.1, it is mentioned that features extracted from C_b component and C_r component have strong correlation so that only one of them is included in our work. To demonstrate this, the average correlation coefficient values and the classification accuracies of different combination of color components are given in Table 3.4. From each component, 162 features from horizontal and vertical directions are extracted. It is observed that the correlation between C_b and C_r component is almost two times the correlation between Y and C_b or Y and C_r. Combing features from Y and C_b together, the classification accuracy was 91.1%. We went further and added features from

Table 3.4 Correlations between Color Components and Classification Accuracies

Color Components	Correlation Coefficients	Feature Dimension	Classification Accuracy
Y		162	85.4%
Cb		162	80.9%
Cr		162	81.0%
YCb	0.4605	324	91.1%
YCr	0.4642	324	90.7%
CbCr	0.9043	324	85.0%
YCbCr		486	91.4%

C_r component in, the accuracy was 91.4%, which is negligible but with the cost of more dimensions (from 324-D to 486-D). Based on these observations, we decided to use only Y and C_b component in this work.

In Section 3.2.1, we explained why it is beneficial to take absolute value of quantized DCT coefficients before calculating the difference array. Here, we compare the classification results of the two cases, i.e., taking absolute values and without taking absolute values to the quantized DCT coefficients. For simplicity, we extracted features from horizontal difference JPEG 2-D array of Y component only. The confusion matrix of not taking absolute values is given in Table 3.5. Table 3.6 displays the confusion matrix with taking absolute values. Comparing these two confusion matrices, we find that the average classification accuracy increased by around 1% (although not very significant) if we take absolute values before calculating difference.

In our work, features are only extracted from difference JPEG 2-D arrays instead of from quantized block DCT coefficient arrays because we believe that by taking difference, the statistical difference can be enlarged. This assumption is empirically verifies in our experimental work too. Table 3.7 gives us the classification result of the

Table 3.5 Confusion Matrix Using Features Extracted from The Difference Arrays of The Original Quantized Block-DCT Coefficient Arrays

	Kodak 6490	**Kodak Z740**	**Nikon D40**	**Nikon 3200**	**Nikon 4600**	**Sony P200**	**Canon 350D**	**Canon SD750**
Kodak 6490	**75.0**	22.0	*	*	*	*	*	*
Kodak Z740	21.0	**76.7**	*	*	*	*	*	*
Nikon D40	*	*	**90.9**	2.1	*	*	*	*
Nikon 3200	*	*	*	**75.3**	20.2	*	*	*
Nikon 4600	*	*	*	20.4	**75.7**	*	*	*
Sony P200	*	*	*	*	*	**97.1**	*	*
Canon 350D	*	*	2.4	2.2	*	*	**91.4**	*
Canon SD750	*	*	*	*	*	*	*	**95.0**

Table 3.6 Confusion Matrix Using Features Extracted from The Difference Arrays of Magnitudes of JPEG 2-D Arrays

	Kodak 6490	**Kodak Z740**	**Nikon D40**	**Nikon 3200**	**Nikon 4600**	**Sony P200**	**Canon 350D**	**Canon SD750**
Kodak 6490	**75.3**	22.8	*	*	*	*	*	*
Kodak Z740	20.2	**78.5**	*	*	*	*	*	*
Nikon D40	*	*	**90.2**	2.4	*	*	3.6	*
Nikon 3200	*	*	*	**78.4**	18.5	*	*	*
Nikon 4600	*	*	*	18.2	**77.9**	*	*	*
Sony P200	*	*	*	*	*	**96.2**	*	*
Canon 350D	*	*	3.2	*	*	*	**91.2**	*
Canon SD750	*	*	*	*	*	*	*	**95.0**

features generated from block DCT coefficient arrays. To make it comparable with Table 3.6, we extracted features from horizontal difference JPEG 2-D array of *Y* component only. The average classification accuracy was 82.8%, obviously lower than the result in Table 3.6, which proved our assumption.

Table 3.7 Confusion Matrix Using Features Extracted from The Original Quantized Block-DCT Coefficient Arrays

	Kodak 6490	**Kodak Z740**	**Nikon D40**	**Nikon 3200**	**Nikon 4600**	**Sony P200**	**Canon 350D**	**Canon SD750**
Kodak 6490	**75.0**	21.8	*	*	*	*	*	*
Kodak Z740	25.4	**72.0**	*	*	*	*	*	*
Nikon D40	*	*	**88.4**	*	2.0	*	2.9	3.1
Nikon 3200	*	*	*	**74.4**	20.5	*	*	*
Nikon 4600	*	*	*	20.6	**74.8**	*	*	*
Sony P200	*	*	*	*	*	**95.4**	*	*
Canon 350D	*	*	3.5	*	*	*	**89.2**	3.0
Canon SD750	*	*	3.0	*	*	*	*	**92.8**

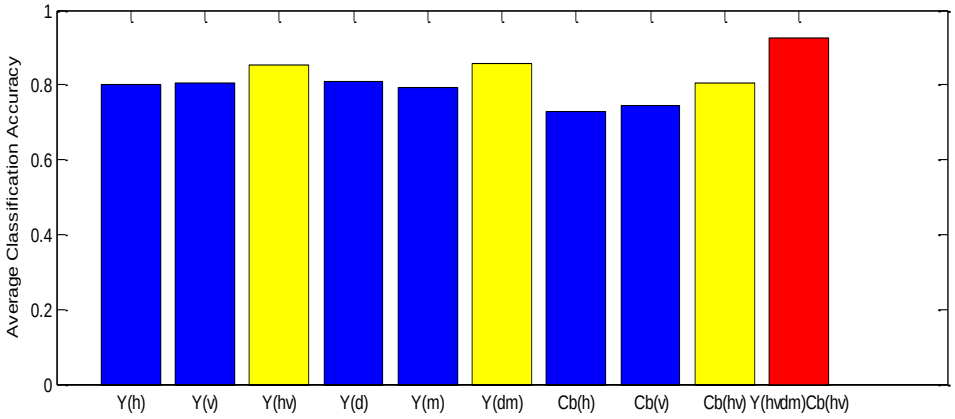


Figure 3.6 Classification ability by directions and color components.

In order to find out how much every transition probabilities calculated along different directions and from different color components contribute to our complete statistical features, we conducted several experiments in which every part of our statistical model were tested separately. The results are shown and compared in Figure 3.6. The horizontal axis in Figure 3.6 represents different parts or combined parts. We use h,v,d,m to denote horizontal, vertical, main diagonal and minor diagonal, respectively. It

is observed that the discrimination power of features generated along four different directions within one color component does not differ much. The performance of features calculated from Y component is generally better than features from C_b component. Hence, the number of features from C_b component in our statistical model is only half the number of features from Y component. The red bar (rightmost) is the final classification result of our proposed model.

3.2.4 Conclusion

Markov transition probability matrix is used in this work to build a statistical feature set that captures statistical difference of difference JPEG 2-D arrays. In total, 486 features are extracted from each image along four directions from Y component and along two directions from C_b component. The results of large-scale experiments have demonstrated the effectiveness of our proposed features.

3.3 Local Binary Patterns in Spatial and Wavelet Domain

In this work, uniform gray-scale invariant local binary patterns (LBP) [63] originally designed for texture classifications were used to generate statistical features for camera model classification. By counting the occurrences of gray-level binary patterns for each pixel against its eight neighbors, 59 LBP features are extracted, respectively, from original red and green color channels in spatial domain, their corresponding prediction-errors and wavelet subband, of each image. Multi-class support vector machines (SVM) were built for successful classification of 18 camera models from the Dresden Image Database [61], a database specifically designed for research in camera identification and other forensic researches. Compared with the results in the literature, the proposed

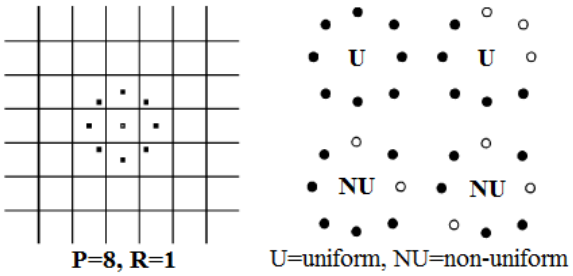


Figure 3.7 (Left) Constellation of neighborhood. (Right) Examples of 'uniform' and 'non-uniform' local binary patterns. This figure is partially borrowed from [63].

statistical features outperformed both the Markov features presented in Section 3.2 and another popular feature set for camera classification [60], and achieved the state-of-the-art performance at the time of publication.

This section is structured as follows. In Section 3.3.1, details of the feature extraction is introduced. Experimental results and some discussions are presented in Section 3.3.2. Summary is given in Section 3.3.3.

3.3.1 Feature Extraction

In [14], local binary patterns (LBP) are of circular neighborhood are introduced. The LBP-encoding of each pixel can be described by

$$LBP_{P,R} = \sum_{p=0}^{P-1} 2^p s\left(g_p - g_c\right) \tag{3.4}$$

where R is the radius of a circularly symmetric neighborhood used for LBP calculation and P is the number of samples around the circle, g_c and g_p denote gray levels of the center pixel and its neighbor pixels, and $s(x)$ is defined as

$$s(x)=\begin{cases}1 & x\geq 0\\ 0 & x<0\end{cases}.$$

In this work, we set R = 1 and P = 8. The constellation of the circular neighborhood we use for local binary pattern calculation is shown in Figure 3.7 (Left).

According to Equation 3.4, gray-level difference is first calculated between the center pixel and its eight neighbors. The difference will then be binary quantized and coded, and in the end transformed to a decimal integer value. After performing the LBP-coding to every pixel in the image, a LBP map will be generated, and the statistics of it will be collected by forming a histogram with a total number of 2^P bins, e.g., 256 bins when P = 8. In addition, in [63], the concept of 'uniform' local binary patterns is introduced. The 'uniformity' is satisfied when the number of binary transitions over a whole neighborhood circle is equal to or smaller than 2. Readers are referred to Figure 3.7 (Right) for some examples. As 'uniform' LBPs occupy the majority of the histogram bins [63], those 'non-uniform' local binary patterns are merged to one bin, thereby suppressing the number of bins from 256 to 59 when P = 8.

Inspired by the fact that quite a lot of image processing algorithms, such as demosaicing, filtering, JPEG compression, are patch-wise (e.g., low-pass filtering with a 5×5 Gaussian mask) implemented inside cameras, it is reasonable to consider that some localized characteristics or artifacts have been generated. These characteristics or artifacts could be effectively captured by the uniform gray-scale invariant local binary patterns. Grayscale invariance is achieved by binarizing the difference between center and neighbor pixels' gray-levels, which to some extent suppresses the influence of image

content. The ‘uniform’ local binary patterns have merged less populated LBP histogram bins into one bin, which is a natural dimensionality reduction advantageous for pattern classification algorithms. Therefore, we propose to use the uniform gray-scale invariant LBP histograms as statistical features to capture camera model characteristics.

As most of the camera image processing algorithms work in spatial domain, a natural choice would be extracting features directly from gray-levels of each color channel in spatial domain. From each color channel, a 59-dimensional (59-D) LBP histogram is generated when $R = 1$ and $P = 8$. Each 59-D LBP feature set are normalized to eliminate the influence of different image resolution. Besides, the same set of LBP features are also extracted from the prediction-error (PE) image. PE image is obtained by subtracting a predicted image from the original image. Considering a 2×2 image pixel block, prediction of a pixel value is achieved by [65]

$$x = \begin{cases} \max(a,b) & c \le \min(a,b) \\ \min(a,b) & c \ge \max(a,b) \\ a + b - \mathrm{c} & \text{otherwise} \end{cases} \tag{3.5}$$

where a is the immediate right neighbor of x; b is the immediate neighbor below x; c is the diagonal neighbor (right and below) of x; and $\hat{x}$ is the prediction value of x. As some image processing algorithms have special treatment at edges and boundaries such as demosaicing and filtering methods, the prediction error image, which is in essence a spatial domain high-pass filtered image that emphasizes edges and boundaries, is another ideal choice to extract features from.

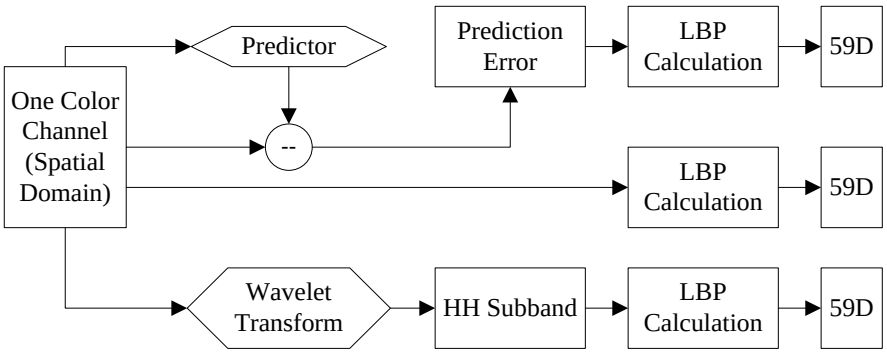


Figure 3.8 LBP feature extraction framework for one color channel.

One of the side-effects of the binary encoding feature of LBP is its insensitivity to monotonic gray-level transform in spatial domain. Although this could be a good feature for some applications, it is not desired for camera model identification, as some in-camera image processing algorithms such as gamma correction has spatial domain monotonic nature and thus the difference of these algorithms could not be captured by our LBP features. To enhance the discrimination ability, in addition to the spatial domain, wavelet domain is considered and we propose to extract another 59-dimensional LBP feature set from diagonal subband (HH subband) of 1st-level Haar wavelet transform.

To conclude, from each color channel, we extract LBP histogram features from original image, its prediction-error 2D array, and its 1st-level diagonal wavelet subband, resulting in a total of 59×3=177 features. The feature extraction framework of one color channel is shown in Figure 3.8. Considering the fact that red and blue color channels usually share the same image processing algorithms, we only use green and red channels. Therefore, the final feature dimensions extracted from a color image is 177×2=354 dimensional (354-D).

3.3.2 Experiments

In this section, some simulation results are presented to demonstrate that our proposed features are able to capture traces caused by different algorithms at a couple of typical image processing tasks insides cameras. We used 20 raw images from Nikon D70 as our basic simulation dataset. All of them are from 'Dresden Image Database'. The dcraw[14] and Matlab are tools we use to mimic the image processing inside cameras. Five different kinds of image processing algorithms are considered, i.e., demosaicing, color space conversion, gamma correction, filtering, and JPEG compression, for each of them, three different algorithms or parameter settings are implemented, displayed in Figure 3.9: (a) Demosaicing algorithms, including bilinear interpolation, VNG: Variable Number of Gradients [107], and PPG: Patterned Pixel Grouping[15]; (b) Color spaces conversion in which images are converted from the original raw space to Adobe RGB[16] and sRGB[17]; (c) Gamma correction, where BT709[18] has gamma=2.4; (d) Image filtering algorithms, including spatial neighborhood averaging, median filtering, and Laplacian of Gaussian (LoG); (e) JPEG compression with QF (quality factor) equals 60, 80 and 100. After feature extraction and projection with linear discriminant analysis, high-dimensional features are projected to two-dimensional space with linear discriminant analysis (note that the two axes have no real meaning). The processing output are clearly clustered according to different algorithms instead of image contents because all the processings are done on the same 20 images, thereby demonstrating the discrimination ability of our proposed features on different in-camera image processing algorithms.

[14] https://www.cybercom.net/~dcoffin/dcraw/ (accessed on November 30, 2016)
[15] https://sites.google.com/site/chklin/demosaic (accessed on November 30, 2016)
[16] http://www.adobe.com/digitalimag/pdfs/AdobeRGB1998.pdf (accessed on November 30, 2016)
[17] https://webstore.iec.ch/publication/6169 (accessed on November 30, 2016)
[18] http://www.itu.int/rec/R-REC-BT.709/en (accessed on November 30, 2016)

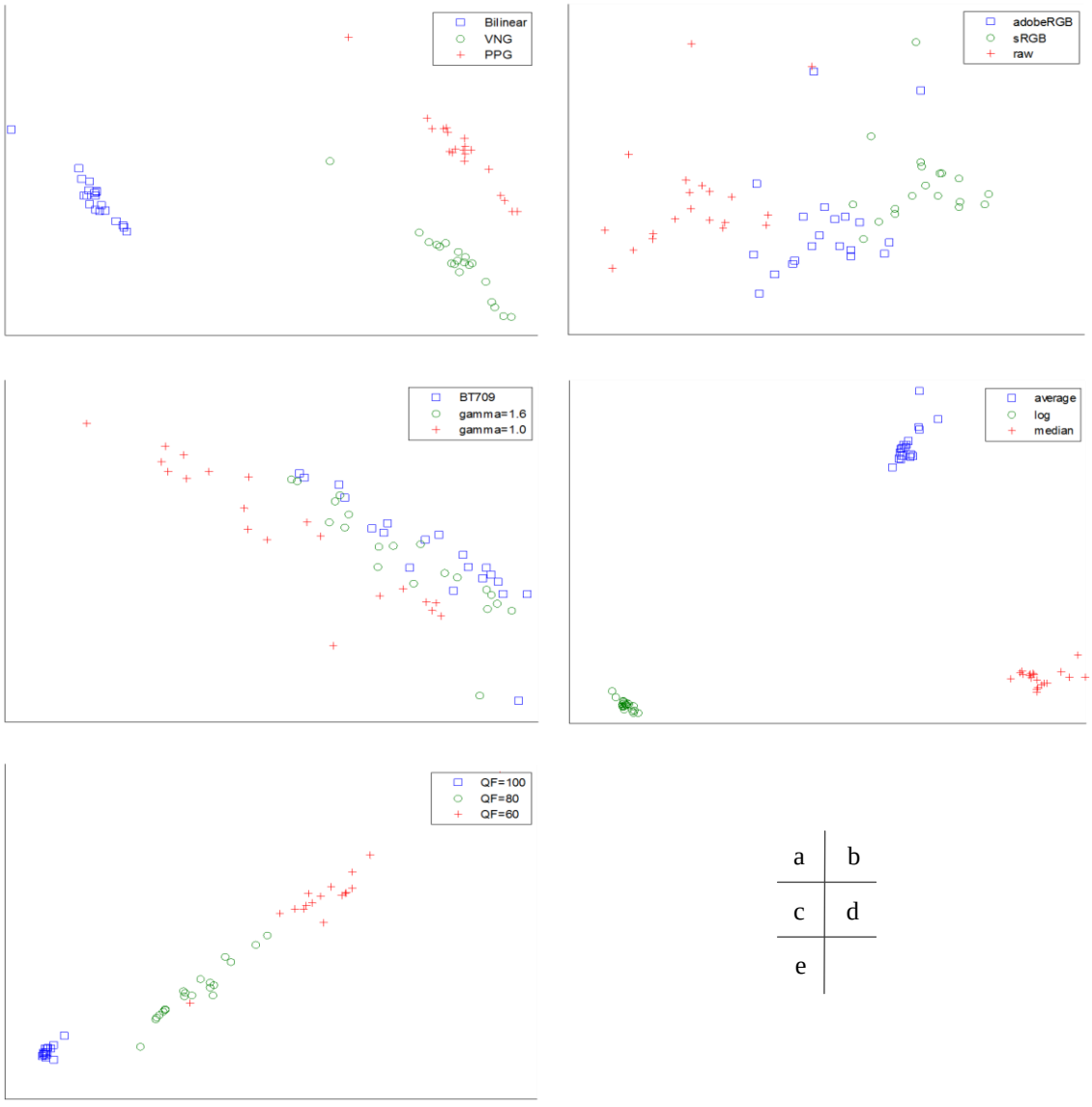


Figure 3.9 2-D projection results from the whole feature set by linear discriminant analysis.

Table 3.8 Experimental Dataset.

Camera Model	# devices	# images	Abbr.
Canon Ixus 70	3	567	CAN
Casio EX-Z150	5	925	CAS
Fujifilm FinePix J50	3	630	FUJ
Kodak M1063	5	2087	KOD
Nikon Coolpix S710	5	925	NIK1
Nikon D70/D70s	2/2	736	NIK2
Nikon D200	2	752	NIK3
Olympus MJU	5	1040	OLY
Panasonic DMC-FZ50	3	931	PAN
Pentax Optio A40	4	638	PEN
Praktica DCZ 5.9	5	1019	PRA
Ricoh Capilo GX100	5	854	RIC
Rollei RCP-7325XS	3	589	ROL
Samsung L74	3	686	SA1
Samsung NV15	3	645	SA2
Sony DSC-H50	2	541	SY1
Sony DSC-T77	4	725	SY2
Sony DSC-W170	2	405	SY3

For large-scale experiments, we picked the same 18 camera models from 'Dresden Image Dataset' as used in [61]. The number of camera devices for each model ranges from 2 to 5. The number of images per model ranges from 405 to 2087. All the images are direct camera JPEG outputs which are captured with various camera settings. Details are given in Table 3.8.

In all of our experiments, multi-class support vector machines (SVM) [66] are trained and used as the classifiers for testing. From the whole dataset, we randomly selected one camera from each model, and used all the images taken by the selected cameras for testing. Images from the rest of the cameras formed the training data. This random selection procedure was performed 20 times for each experiment. Involving images from more than one camera of each model (except those have only two cameras)

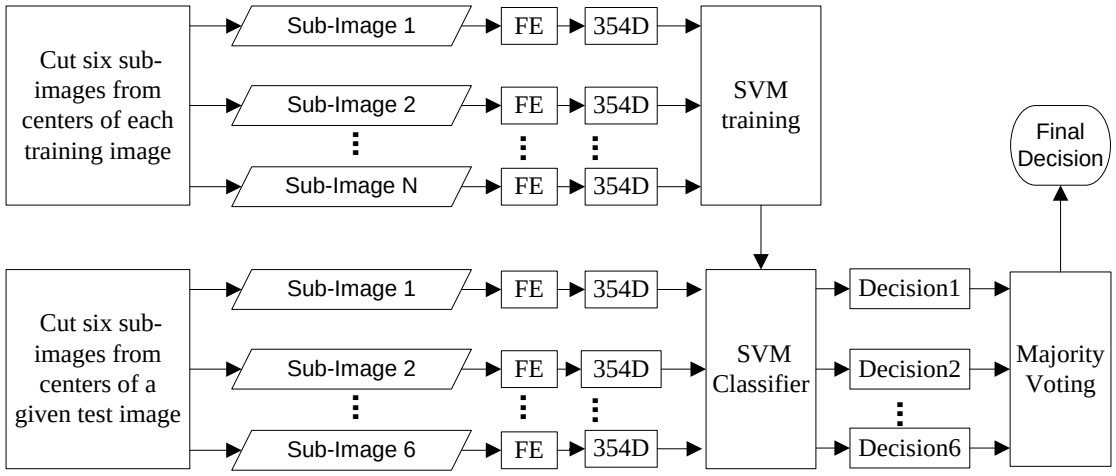


Figure 3.10 Block diagram of training and testing stages. FE: feature extraction.

for training greatly reduced the chance of overfitting to a specific camera instead of a camera model. Using the cameras that were not involved in the training procedures for testing made the experiments more practical.

In each random split, images for both training and testing were cut into six sub-images from centers. The final decision in the testing stage was made for each image by majority voting of the six individual decisions. Ties were broken by random assignments. This cropping and voting procedure not only increased the number of samples for training, but also brought robustness against the regional anomalies in testing images. A block diagram is shown in Figure 3.10 which includes both the training and testing stages (only one image is shown in the testing stage).

The classification results with our proposed features are reported in Table 3.9, which provides the confusion matrix averaged over 20 splits. The average identification accuracy reached more than 98% for 18 camera models. Note that in [61], average accuracy of 96.42% is reported using the same camera models in the 'Dresden Image Database'. Although the proposed method identification accuracy was higher by only

Table 3.9 Average Confusion Matrix (in %).

average accuracy = 98.1		predicted camera model																	
		CAN	**CAS**	**FUJ**	**KOD**	**NIK1**	**NIK2**	**NIK3**	**OLY**	**PAN**	**PEN**	**PRA**	**RIC**	**ROL**	**SA1**	**SA2**	**SY1**	**SY2**	**SY3**
actual camera model	**CAN**	100	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	CAS	–	99.52	–	–	–	–	–	–	–	0.48	–	–	–	–	–	–	–	–
	FUJ	–	–	100	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	KOD	–	–	–	100	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	NIK1	–	–	–	–	100	–	–	–	–	–	–	–	–	–	–	–	–	–
	NIK2	–	–	–	–	–	100	–	–	–	–	–	–	–	–	–	–	–	–
	NIK3	–	–	–	–	–	2.39	97.58	–	–	–	–	–	–	–	–	–	–	–
	OLY	–	–	–	–	–	–	–	100	–	–	–	–	–	–	–	–	–	–
	PAN	–	–	–	–	–	–	–	–	100	–	–	–	–	–	–	–	–	–
	PEN	–	–	–	–	–	–	–	–	–	100	–	–	–	–	–	–	–	–
	PRA	–	–	–	–	–	–	–	–	–	–	100	–	–	–	–	–	–	–
	RIC	–	–	–	–	–	–	–	–	–	–	–	100	–	–	–	–	–	–
	ROL	–	–	–	–	–	–	–	–	–	–	–	–	100	–	–	–	–	–
	SA1	–	–	–	–	–	–	–	–	–	–	–	–	–	100	–	–	–	–
	SA2	–	–	–	–	0.28	–	–	–	–	–	–	–	–	–	99.72	–	–	–
	SY1	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	93.76	–	6.24
	SY2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	100	–
	SY3	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	25.10	–	74.90

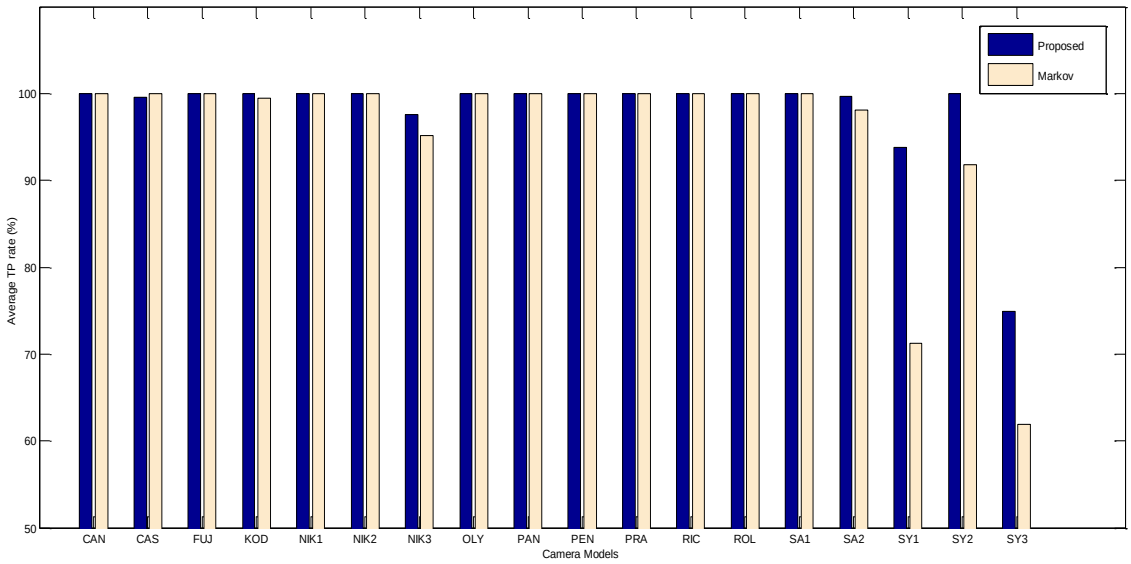


Figure 3.11 Comparison of classification results using LBP-based features and Markov-based features.

about 1.5%, it actually reduced the error rate by more than 40%. For comparison, we also tested the Markov features proposed in Section 3.2. Figure 3.11 displays the results by comparing the classification accuracy model by model between features proposed in this work and those proposed in Section 3.2 using a bar graph. We can see that our proposed LBP-based features outperform the Markov features for most of the camera models.

Although the average detection rate is high, we note that the detection rates for Nikon D200, Sony H50 and Sony W170 are 97.58%, 93.76% and 74.90%, respectively, which are relatively low. From Table 3.8, we can see that the number of individual cameras of these three camera models is two, which means only one camera per model is involved in training. In this case, there exist two possibilities that cause the low detection accuracies for these three models. Either the LBP features could not capture model characteristics for these three camera models well, or they actually capture more individual camera characteristics. In order to clear up this issue, we did some additional

Table 3.10 Confusion Matrix between Two Nikon D200 Cameras

average accuracy = 79.81		**Predicted**	
		D200-1	**D200-2**
Actual	**Nikon D200-1**	**78.68**	21.32
	Nikon D200-2	19.07	**80.93**

Table 3.11 Confusion Matrix between Two Sony H50 Cameras

average accuracy = 53.64		**Predicted**	
		H50-1	**H50-2**
Actual	**Sony H50-1**	**54.93**	45.07
	Sony H50-2	48.32	**51.68**

experiments by classifying images produced by cameras of the same model; these experiments could test within-model discrimination ability of our features. A higher identification rate here implies more chance of overfitting to specific camera devices; the ideal classification rate would be random guess. Results are given in Tables 3.10 – 3.12. In Table 3.10, it is shown that the detection accuracy between two Nikon D200 cameras is almost 80%, which is much higher than random guess (50%). Therefore, overfitting could be the cause of lower detection accuracy. This could possibly be solved by adding more devices of Nikon D200 to make the classifiers more difficult to overfit to specific cameras. For the other two Sony camera models, results in Tables 3.11 and 3.12 demonstrate low intra-model similarity, thus eliminating the possibility of overfitting. Therefore, we can conclude that our feature set could not reliably identify those two Sony camera models.

The testing results of discrimination abilities of LBP features extracted from spatial domain, prediction-errors, and wavelet subbands of both red and green channels

Table 3.12 Confusion Matrix between Two Sony W170 Cameras

average accuracy = 58.21		**Predicted**	
		W170-1	**W170-2**
Actual	**Sony W170-1**	**60.57**	39.43
	Sony W170-2	40.45	**59.55**

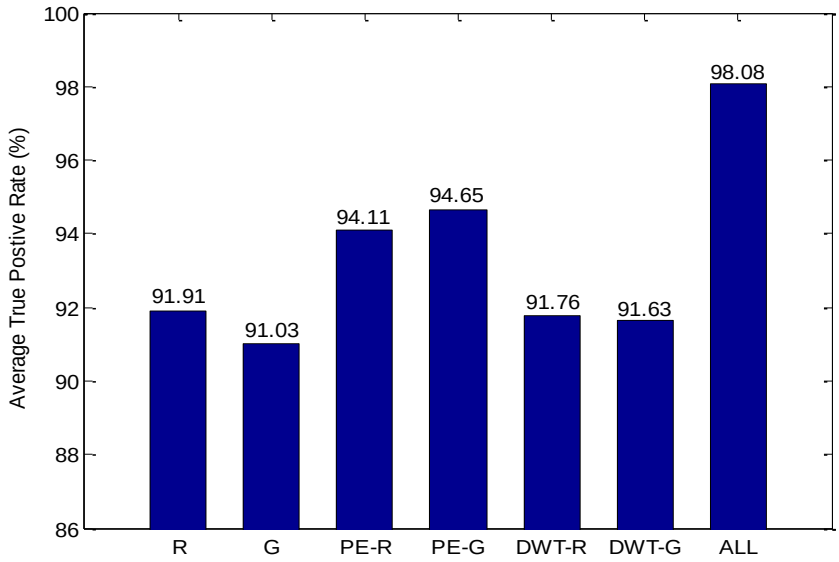


Figure 3.12 Classification accuracy using LBP features extracted from different image 2-D arrays and the combined features proposed.

are shown in Figure 3.11, from which we can see that the combined LBP features do improve the overall identification performance.

3.3.3 Conclusion

We propose in this work the uniform gray-scale local binary patterns as features for camera model identification. By combining features extracted from the original image, its prediction-error image, and the HH subband of the image's 1st level wavelet transform, the proposed scheme has demonstrated improved performance in camera model classification.

3.4 Discussion

For camera model classification, two effective feature sets have been proposed in this chapter. These features, particularly the LBP-based, had the best classification performance at the time when they were proposed. Since these researches have been completed in the early stage of the Ph.D., some more recent studies [108] have shown that feature subsets carefully selected from the SRM [12] originally proposed for steganlaysis have marginal performance improvement over our proposed LBP-based features. Nevertheless, all the feature-based methods can be and will be replaced by CNN-based methods (see Chapter 2) for camera model classification. In fact, some results have emerged recently that shows the potential power of CNN-based methods [109, 110]. It would be interesting future works to apply the sophisticated CNN, proposed in Chapter 2 for steganalysis, to camera model classification.

CHAPTER 4

IMAGE TAMPERING DETECTION IN REAL WORLD

4.1 Introduction

Due to the ever increasing power of image editing software, such as Photoshop and Gimp, creating fake images have never been easier. This could give rise to serious problems whenever images are treated as important evidence, or published by mass media to disseminate important information, as one can never take for granted the authenticity of those images. Figures 4.1 and 4.2 show two examples of famous forgery. In 2004, a picture of John Kerry and Jane Fonda at an anti-war rally during the early 1970's surfaced on the Internet for some political motivations, which is the left image of Figure 4.1. It was reported later that this picture was created by merging the center and right images in Figure 4.1. Figure 4.2 (Left) is an image about Israel air striking Beirut, Lebanon in August 2006. This image was later found altered by the photographer and the original authentic image is displayed in Figure 4.2 (Right). Compared with the authentic image, the altered image has made smoke darker by some image processing software. This forged picture caused Reuters to withdraw 920 pictures taken by the photographer from sale. By searching through the internet, we can find such kinds of 'fake' photos everywhere. Our society is in urgent need of advanced forensic technology to catch the 'image tampering' and recover the credibility of digital images in real-world.

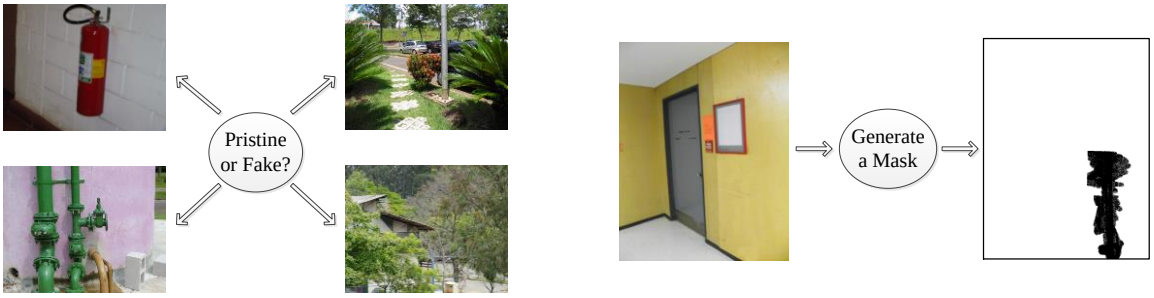
Figure 4.1 (Left) A spliced image. (Center and Right) The two original images that formed the sliced image.

Source: http://www.snopes.com/photos/politics/kerry2.asp

Figure 4.2 (Left) An altered image. (Right) The original image.

Source: http://news.bbc.co.uk/2/hi/middle_east/5254838.stm

Blind and passive digital image tampering detection [67, 68, 92] (tampering detection in short), as one of the biggest research areas in image forensics, aims at finding evidence of image forgery without relying on any side information or watermarking, as they can be either unreliable or not available. Its main task is to decide if an image under investigation has been tampered or not, and if possible, to locate the tampered regions. Detection of tampered images can be considered as basic forensics, while locating the tampered regions is considered more advanced function that can reveal more important evidence, such as telling what object has been added in, or something of certain size and at certain location has been removed.



(a) (b)

Figure 4.3 An illustration of (a) tampering detection and (b) tampering localization.

In the last years, lots of tools and algorithms have been developed by researchers and forensics experts to interpret the authenticity of digital images. In Digital Image Forensics Database[19], over 600 papers have been published over the past ten years, and most of them are related to tampering detection. However, the diversity of the sub-fields in tampering detection, and the fact that existing public database [69] overlooks real-world conditions, call for a practical benchmark and common comparison protocol of published algorithms.

To actively move the research on image tampering detection ahead, the Technical Committee of Information Forensics and Security at IEEE Signal Processing Society had successfully organized a competition on Image Tampering Detection in the summer and fall of 2013[20]. The competition was worldwide and consisted of two phases: Phase-1 and Phase-2.

In Phase-1, 1500 labeled (authentic or fake) training images were provided for the participated teams to build models, which would then be used to predict the labels for 5713 testing images to evaluate the performance of the models, as illustrated in Figure

[19] http://www.cs.dartmouth.edu/~farid/dfd/index.php/publications (accessed on November 30, 2016)
[20] http://ifc.recod.ic.unicamp.br/fc.website/index.py (accessed on November 30, 2016)

4.3 (a). Among the 1500 training images, there are 1050 authentic images (the negative class) and 450 tampered images (the positive class). This is an imbalanced binary (two-class) classification problem. The evaluation metric used was the balanced accuracy defined as

$$accuracy = \frac{TNR + TPR}{2}, \tag{4.1}$$

where *TNR* denotes true negative rate obtained by dividing total number of correctly classified authentic images by the total number of authentic images, *TPR* stands for true positive rate which can be obtained by dividing the total number of correctly classified fake images by the total number of fake images. Our team got the runner-up prize with the balanced accuracy of 93.72%, lost by merely 0.48% to the first prize winner.

Compared with the image-level classification task required in Phase-1, Phase-2 was a more challenging task. A total of 700 tampered images (no authentic images) with various spatial resolutions were given by the organizer. The participants were asked to submit a binary mask for each image to point out the tampered region pixel-wisely, as illustrated in Figure 4.3 (b). This is a binary classification problem to output binary prediction for each pixel. Tampering localization performance is measured by averaging *F-score* $\in \mathbb{R}^{[0,1]}$ across all the testing images. The *F-score* can be expressed as

$$F_{score} = 2 \times \frac{precision \times recall}{precision + recall}, \tag{4.2}$$

where

$$precision = \frac{TP}{TP + FP} \quad , \quad recall = \frac{TP}{TP + FN}.$$

Here *TP* is the total number of tampered pixels that are correctly detected in the image, FN is the total number of miss-detected tampered pixels, *FP* is the total number of pixels falsely detected as tampered. Hence, *precision* is the ratio that a pixel detected as tampered is truly a tampered pixel, and *recall* is the ratio that a tampered pixel is detected. The author of this dissertation again got the runner-up prize, however, with an average *F-score* of only 0.2678. The winner achieved 0.4071 which is also far from satisfactory comparing with the results from Phase-1. The results of Phase-2 indicate unsatisfactory localization performance of tampering detection technologies when facing real-world and modern forgeries, and encourage more practical and valuable works in the future to boost the performance of tampering localization.

In this chapter, we first report what we have tried in Phase-1 of the competition and the final solution, i.e., an image-level tampering detection method based on advanced statistical features for advanced steganalysis with some modification to drastically reduce feature dimensionality while boosting the detection accuracy. Then, a fast block-based copy-move detector exploiting PatchMatch for block matching was proposed. This block-based copy-move detector, together with a very basic feature-based copy-move detector using scale-invariant feature transform (SIFT), form our main solution to forgery localization in Phase-2 of the competition. Because of the tight competition schedule, many of the existing advanced techniques have not been tested, it is expected that by

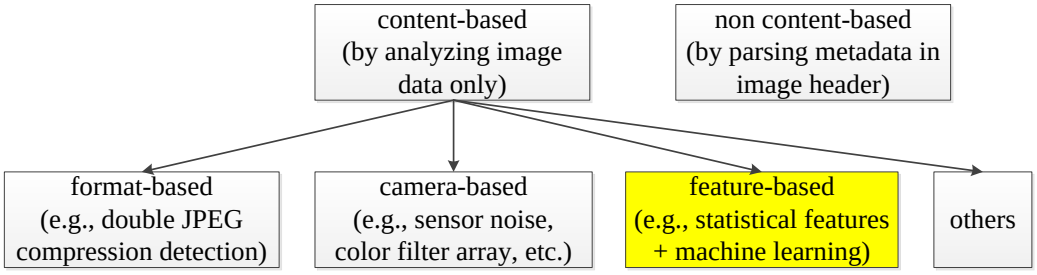


Figure 4.4 Approaches to tampering detection.

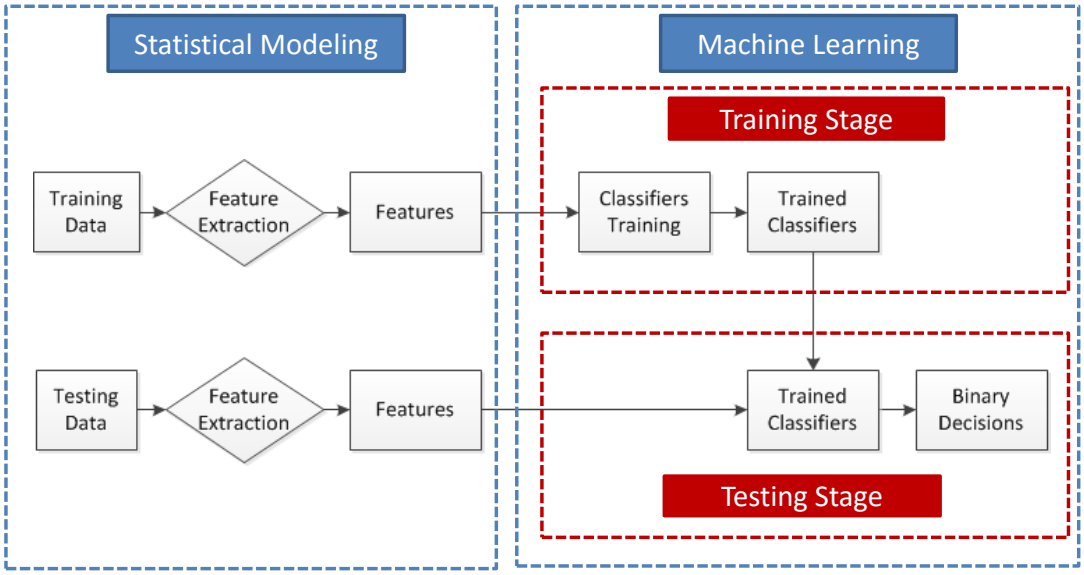


Figure 4.5 A general framework of statistical feature-based tampering detection.

exploring the newest technologies developed in image tampering detection as well as a summary for the past research along this direction, the results can be further improved.

The rest of this chapter is organized as follows. In Section 4.2, we discuss our solution to image-level tampering detection in Phase-1 of the competition. In Section 4.3, given that all the provided images had been tampered, the solution to pixel-level tampered region localization is presented, which corresponds to Phase-2 of the competition. The methodology comparisons with the winner are given in Section 4.4.

4.2 Solution to Phase-1: Tampering Detection

Potential approaches to solve the tampering detection problem in Phase-1 of the competition include format-based [70-77, 102], camera-based [78-81], statistical feature-based [82-85], etc. In our solutions, statistical feature-based methods were used to tackle this binary classification problem, because of their less limited applicability. The assumption is that tampering operations result in unnatural pixel statistics and possible inconsistency along the tampered regions. Feature-based methods focus on designing suitable features to train the following machine learning based classifiers which rely heavily on mathematical optimization. A general framework of statistical feature-based tampering detection is illustrated in Figure 4.5. Three feature sets as input of classifiers have been developed in a row by our team. It turns out that the best performer among the three was a subset of high-dimensional feature set originally designed and used for steganalysis. Support vector machines (SVM) and the ensemble classifiers of fisher linear discrimant (FLD-ensemble) [30] have been adopted for classification. Details of the three feature sets and their performance on testing data are covered in Sections 4.2.1, 4.2.2, and 4.2.3.

4.2.1 Pure LBP-based features

Local Binary Patterns (LBP) [63] was proposed as an effective texture classification technology, and has been utilized for face recognition and image forensics, including steganalysis [86] and camera model classification [87]. In Phase-1 of the competition, LBP were used to model original pixel values in spatial domain, and the LBP histograms extracted are used as the statistical features. Note that most of the images provided by the

Table 4.1 Confusion Matrix (in %) Using Uniform LBP

Accuracy = 71.4	**Predicted Negative**	**Predicted Positive**
Actual Negative	**94.37**	5.63
Actual Positive	51.33	**48.67**

Table 4.2 Confusion Matrix (in %) Using Original LBP

Accuracy = 91.2	**Predicted Negative**	**Predicted Positive**
Actual Negative	**94.91**	5.09
Actual Positive	12.47	**87.53**

organizers of the competition were color images; before we start the feature extraction process, color images were transferred to grayscale images.

In the training set, 1050 images in 'authentic' (negative) class and 450 images in 'fake' (positive) class were provided by the organizers. SVM with polynomial kernel served as the classifiers for the experiments in this section. Balanced accuracies on the training set are reported in Tables 4.1 and 4.2. In Table 4.2, all of the 256-dimensional (256-D by assuming number of neighbors equals eight) LBP features were used, while in Table 4.1, only the so-called 'uniform' [63] features of LBP were considered and the dimensionality was reduced from 256-D to 59-D. However, a significant performance drop was observed comparing with the original 256-D. Therefore, we chose to use the classifier trained by 256-D LBP to predict the testing dataset. The accuracy feedback provided by the online system was around 85%. This first trying was encouraging. Later we got better results in our attempts.

4.2.2 Hybrid Feature Sets

Encouraged by the initial success with the LBP and SVM, it is natural to enhance the performance by building more advanced and hence complicated statistical models. In [84]

Table 4.3 Confusion Matrix (in %) Using The Combined Feature Sets

ACC = 94.9	**Predicted Negative**	**Predicted Positive**
Actual Negative	**97.26**	2.74
Actual Positive	7.47	**92.53**

and [85], the moments of 1-D and 2-D characteristic functions (moments-based) and probability elements in Markov transition probability matrices (Markov-based) extracted from multi-size block DCT coefficients of images are combined together. This statistical feature set has previously achieved excellent results on tampering detection in the Columbia dataset [69], another existing dataset for splicing detection. Inspired by the success of [84] and [85], we managed to fuse various feature set by vector concatenation and came up with a diverse and more powerful feature set. Specifically, besides LBP-based features, moments-based and Markov-based features were calculated from coefficients arrays generated by block-DCT transforms with block sizes equal 2×2, 4×4 and 8×8, and the Local Derivative Patterns (LDP) [88] which improved LBP by capturing directional changes of derivatives of neighboring pixels against central pixel were also included. As the feature size is large, only horizontal and vertical directions of LDP were considered, resulting in 2×256 = 512-D features. Components of involved in our hybrid feature model are summarized below:

- 256-D basic LBP-based features calculated from original spatial domain.
- 512-D LDP-based features calculated from original spatial domain.
- 168-D moments-based features calculated from original image and multi-block DCT 2D arrays.
- 972-D Markov-based features calculated from multi-block DCT 2D arrays.

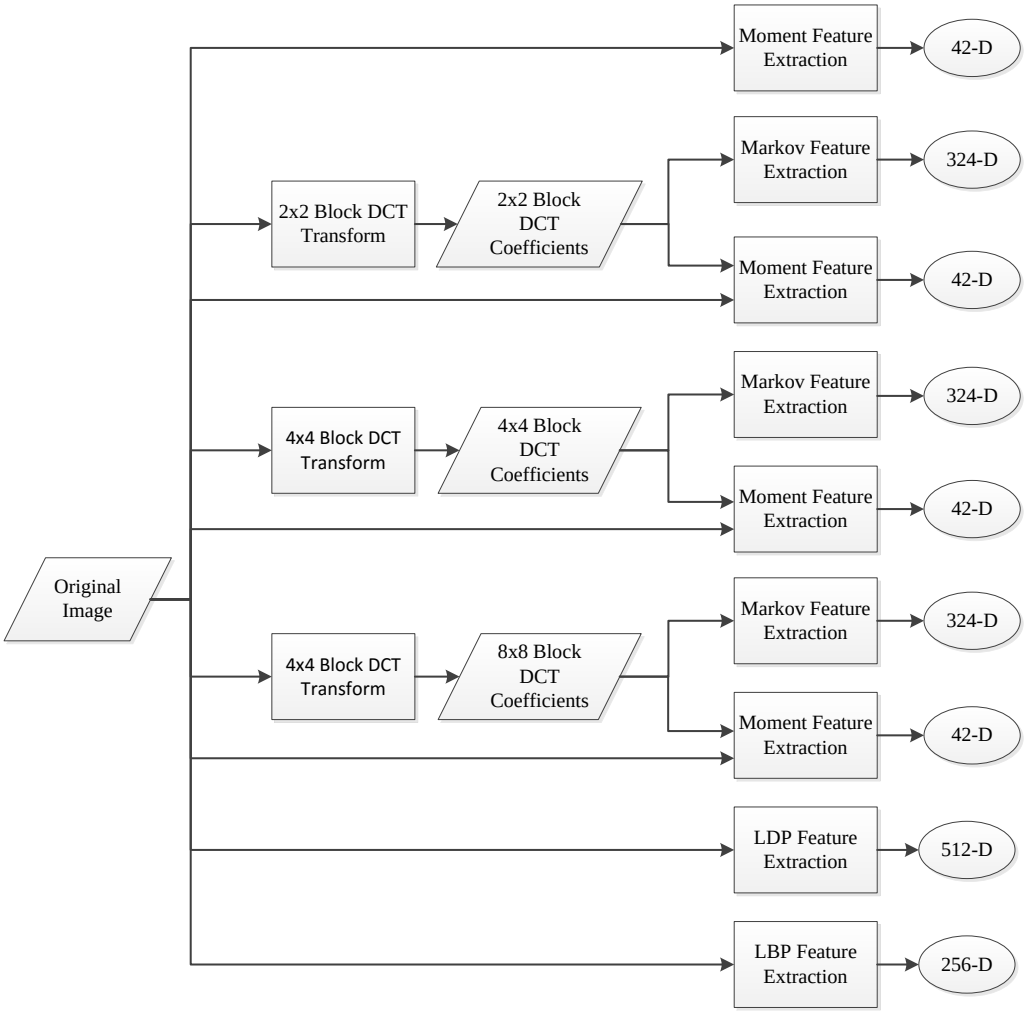


Figure 4.6 Feature extraction framework of the proposed hybrid feature set.

Figure 4.6 shows the block diagram of this statistical feature extraction. The final feature vectors were combined through vector concatenation and reached dimensionality of 1,908-D. We used the same training process as in our first attempt using LBP-based features alone. Result is shown in Table 4.3. The training accuracy boosted from 91.2% to 94.9%, which gives us confidence that the accuracy should also improve for testing set. However, surprisingly, the feedback result is only around 81%. Most likely, this abnormality was caused by a bug in testing score calculation which was reported by some of the participants and fixed later on by the organizers.

4.2.3 SRM-based Features

Through the experiments reported above, we realized that features derived from spatial domain could have more classification capability for tampering happened in spatial domain. In [21], Fridrich and Kodovský proposed to combine high-order co-occurrence probabilities extracted from various noise residuals (obtained by high-pass filtering) of the original images to break the most secure spatial-domain steganographic algorithms, hence it is often called the spatial rich model (SRM). While steganalysis and tampering detection are different research areas, it is recognized that both steganalysis and feature-based tampering detection rely on the change of statistics of pixel-neighborhood caused by secrete message embedding and the tampering operation, respectively. Previous works [84, 85] have shown that methods designed for steganalysis can work well for tampering detection, given that the corresponding classifiers are trained by the samples of image tampering. Furthermore, the statistical features proposed in [12] contain the desired high-order statistical features (co-occurrence of four consecutive pixels is considered), i.e., more powerful statistical modeling in the spatial domain. Hence, we applied the SRM features in this competition, more accurately, the SRMQ1 [12] feature set which has only one-third of the features compared with the full-version of SRM, even so, the total feature dimensionality reaches 12,753-D.

Some changes were made in the experimental settings this time. From the results in Tables 4.1 to 4.3, we noticed that the accuracies for the positive and negative classes were imbalanced. Very likely the imbalance was caused by not having enough training data for the positive class (tampered images). As there was no mandatory requirement on using only the provided training set for training purpose, we added all the 700 testing

Table 4.4 Average Validation Error Rate (in %) Using SRMQ1 Feature Set

Average Error	False Negative	False Positive
3.66	3.83	3.49

images in Phase-2 of this competition to positive class for training because all of them were claimed to be fake. Therefore, we had more data for positive class and the training set became much more balanced — 1050 negative samples and 1150 positive samples. The FLD-ensemble classifiers [22] used in [21] for classification was also inherited to replace the SVMs because of the higher feature dimensionality.

The FLD-ensemble classifiers require a validation set to optimize two hyperparameters. In all the following experiments, we set the training/validation ratio to 0.8/0.2 of the training set, and the number of random training/validation splits to 13. The FLD-ensemble classifiers also require equal number of training sample for both classes; hence, data ensemble was applied. As there were 100 more image data in positive class, a random selection of 1050 out of 1150 positive samples was carried on before the start of training process. The data ensemble made sense because the competition adopted balanced accuracy for evaluation, and therefore, there was no reason to have bias on the number of training sample for either class. The number of data ensemble we made was also 13, so during the training process, 13×13 = 169 classifiers were built from the training set and ready to be applied to the testing set containing 5,731 images. The final decisions made on the testing images were obtained by majority voting the 169 decisions. Validation errors were generated along with the classifier training process. Using the 12,753-D SRMQ1 feature set, the average validation error rates calculated by $0.5 \times (FPR + FNR)$, where FPR and FNR stand for the false positive rate and false negative rate,

Table 4.5 Average Validation Errors (in %) and the STDs for Every Residual Type in SRMQ1

Residual Type	Dimension	STD	AVG ERR	FN	FP
S1_minmax	3250	6.30	5.03	3.97	6.08
S2_minmax	1625	3.60	4.47	4.34	4.60
S3_minmax	3250	4.97	4.42	4.44	4.39
S3x3_minmax	1300	3.45	4.38	4.17	4.59
S5x5_minmax	1300	3.71	4.37	3.49	5.24
S1_spam	338	2.99	6.59	6.61	6.56
S2_spam	338	2.63	6.40	6.98	5.82
S3_spam	338	2.75	5.59	5.14	6.04
S35_spam	338	2.88	6.77	6.30	7.25
S3x3_spam	338	2.54	5.26	4.39	6.14
S5x5_spam	338	2.73	5.79	4.97	6.61

Table 4.6 Average Validation Errors (AVG ERR) (in %)

	AVG ERR	FN	FP
without S3x3_minmax22h	3.92	3.86	3.97
without S3x3_minmax22v	3.97	4.02	3.92
without S3x3_minmax24	3.76	3.97	3.55
without S3x3_minmax41	4.50	3.97	3.55

equaled 3.66%, equivalent to 96.34% in accuracy, as shown in Table 4.4, and the online feedback testing result reached 91.7%.

So far, the full SRMQ1 feature set works quite well. In spite of this success, there was still doubt that some features in the SRMQ1 might have negative contribution, after all, operation made on images with tampering and steganography were different, thus, the original features designed for steganalysis might not be optimal for tampering detection. Having realized this issue, we took one step further and selected a subset of the 12,753-D SRMQ1 feature set, aiming at improving accuracy. Unlike general feature selection that

element-wise select a subset, our feature selection worked on group of features to reduce the complexity. Steps of our feature selection are roughly described below:

1. Divide the whole set of SRMQ1 features into groups based on the residual types.
2. Perform experiments to find validation errors and the standard deviations (STD) for every residual type calculated on training data.
3. Select and combine feature subsets by simultaneously considering validation errors and the STDs.

When forming feature subsets, we basically followed the residual types, i.e., first order (S1), second order (S2), third order (S3), edge 3×3 (S3×3), edge 5×5 (S5×5), and 3×3, 5×5 spam (S35_spam). Features calculated from 'spam' and 'minmax' residuals were considered separately. For details of the residual types, please refer to [12]. The reason we included the STDs into our feature selection was that all of the features generated were co-occurrence probabilities which was basically co-occurrence histogram bins. Since the histogram bins within some residuals were very non-uniformly distributed, and some even had a lot of empty bins, the statistics could be less stable. As the means of each residual type were equal because of the normalization, it was natural to use the standard deviation to measure the uniformity of co-occurrence histograms. Here we assumed that lower STD implied more uniform distribution and hence preferred.

Table 4.5 shows the validation error rates and STD corresponding to each residual type. By simultaneously considering these two factors, we chose three groups: S3_spam, S3×3_spam and S3×3_minmax, and the total feature dimensionality thus was reduced to 1,976-D. Since the feature size is still high in S3×3_minmax (1,300D), we performed backward selection on all the four co-occurrence matrices included, i.e., each time we removed one 325-D co-occurrence histogram based on the validation results and STDs. In the end, only one 325-D co-occurrence histogram (S3×3_minmax24) was removed.

Details of the results in the backward selection are reported in Table 4.6. The total dimensionality now has been reduced to 1,651D from 1,976-D. This 1,651D served as our final feature set to build the classifier for Phase-1 of this competition. To make it clear, the final set we use is S3_spam (338D) + S3×3_spam (338D) + S3×3_minmax22h (325D) + S3×3_minmax22v (325D) + S3×3_minmax41 (325D) = 1,651D. The online feedback testing accuracy is around 93.8% – 94.0%. There is about a 2% increase compared with the whole 12,753-D SRMQ1 feature set. Although this may not be the optimal subset, it is the best we can do within the limited period of time.

4.3 Solution to Phase-2: Tampering Localization

In Phase-2 of the competition, the participants were required to locate the tampered regions pixel-wisely. Before starting the research, an analysis was made based on the training data about the major tampering methods. By observing the fake images in training data and the corresponding ground truth masks, the tampering methods could be classified into two major categories: copy-move (tampered regions replaced by other regions from the same images) and splicing (tampered regions replaced by regions from other images). Note that these two categories might not cover all the tampering cases, but they were definitely the main stream.

Once we limited the tampering methods to work with, the next step was to design corresponding forensic methods. The major idea of our algorithm design was to break big problem into smaller problems and design corresponding algorithms to solve each small problem. Each of the designed algorithms worked independently and the last step was to fuse the outputs.

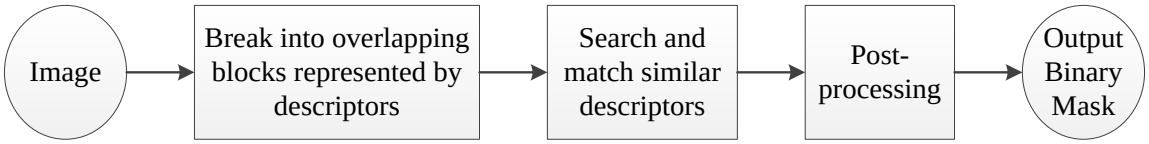


Figure 4.7 The general framework of copy-move forgery detection.

Table 4.7 Copy-Move Cases vs. Forensic Methods

Forensic Method	Copy-Move Cases					
	Smooth (wo)	**Texture (wo)**	**Object (wo)**	**Smooth (w)**	**Texture (w)**	**Object (w)**
Hamming Distance of LBP	YES	YES	YES	N/A	NO	NO
Euclidian Distance of SIFT	NO	YES	YES	N/A	YES	YES

'YES' — The forensic methods can tackle this copy-move case.
'NO' — The forensic methods cannot tackle this copy-move case.
'N/A' — the copy-move case not considered
'w' — with further processing
'wo' — without further processing.

The copy-move problem could be separately into three categories based on the content of copied regions, namely, smooth areas, texture areas, and objects. The tampered regions could be further processed, e.g., through scaling and rotation. To solve the problem that tampered regions were directly copy-moved without any further processing, similarity is compared between image blocks (patches). We proposed a new distance measure that worked by counting the total hamming distance of LBPs calculated from corresponding pixels inside patches. Similarity was measured based on the count of hamming distance. This distance measure has the advantage that as long as there is no other processing, copied regions can be detected even in smooth regions due to camera sensor noise, not to say in textured region and objects. However, it would not work in the scenario that the tampered regions were further processed. Fortunately, based on our observation and study of popular image editing software, smooth areas, most likely, have

not gone through any further processing. As most of the images have more than 1024×768 pixels, the searching process could be rather slow. To speed up the searching process, we adopted the PatchMatch algorithm [89, 90] for efficient copy-move detection.

For copy-move cases that had involved further processing on non-smooth tampered regions, we simply performed a brute-force matching between the scale-invariant feature transform (SIFT) [15] features with Euclidean distance as the similarity measure.

Table 4.7 shows the different copy-move cases versus our forensic methods. As we entered the competition rather late, we have only designed algorithms for copy-move tampering localization. Splicing localization has to be future work.

4.3.1 PatchMatch and Hamming Distance of LBP Blocks

Copy-move forgery detection is one of the most popular topics in image tampering detection. The solutions are quite similar and all of them are based on the nature of this tampering technique — to find and alarm regions that are similar within an image. There are three key elements in almost all of these algorithms, i.e., descriptor generation from each block (patch), similarity (distance) measure between the descriptors extracted from two blocks, fast searching algorithm for block matching. The approaches to descriptor generation have the main impact on the accuracy of block-matching as well as some influence on searching speed. The block searching and matching algorithms have also some impact on accuracy, but the major concern is to speed up the searching process of the duplicated regions. The framework of copy-move detection is displayed in Figure 4.7. In [91], an evaluation was given on all the existing copy-move methods proposed by year 2013 The benchmark results given in Table V of [91] shows that the average descriptor

generation time is about one hour for a single image with average size of roughly 3000×2300, and the average matching time is about one and half hour. Therefore, speedup is required for copy-move detection; otherwise they cannot find practical use for real world images which may have even more pixels. Before we introduce our descriptor generation method and distance measure, we first introduce the PatchMatch algorithm, which had served as our tool for block matching.

PatchMatch is initially proposed as a method that bring revolutionary speedup for matching regions in image A with the most similar regions in a different image — image B. According to Table 1 in [89], PatchMatch is dozens of times faster than the popular tree-based searching method. The algorithm of PatchMatch contains mainly three steps:

1. Determine a patch size around pixels. Typical patch sizes can be 3×3, 5×5, …, 15×15, depending on the applications.
2. Randomly permute patches in image B, and assign each patch in image A with a patch in image B.
3. Loop through patches in image A: for each patch in image A, check its neighboring patches to see if they have found a more similar patches in image B, if so, look into the corresponding patch and its surrounding patches in original image B (not-permuted), and perform update.

Steps 3 are usually performed multiple times with different sequence of looping until convergence.

The adaptation of the original PatchMatch algorithm to the copy-move detection is straightforward. In copy-move detection, we need to find duplicated regions within same images; the essential step is to find for each region in a given image the most similar region within the same image. Hence, the main PatchMatch algorithm was inherited, with only three slight changes: 1) in copy-move detection, image B is the same

as image A; 2) since the most similar patches are themselves, a minimal similarity threshold was set to prevent self-assignment; 3) for each test image, the PatchMatch algorithm was run multiple times independently with patch sizes of 5×5, 7×7, 9×9 and 11×11, the output of each of them were fused.

Now we discuss the descriptors and the similarity (distance) measure. In the original PatchMatch papers [89, 90], the average Euclidean distance of all the pixel values in corresponding patches is used as similarity measure. This similarity measure was expected to create a lot of false positives in smooth areas because the Euclidean distances between patches in smooth regions are all very close. As Barnes *et al.* in [89, 90] mentioned that any distance can be used to replace the Euclidean distance, we proposed to encode every pixel in image patches with LBP, which is an 8-bit binary string generated by comparing the values of the pixel and its eight neighbors one by one, and the similarity measure we used was the Hamming distance between two LBPs of corresponding positions of two image patches. Unlike the most common use of LBP that convert the encoded binary strings to decimal values and calculate the histogram of those decimal values in the whole image, we used the LBP coded map directly so as to keep the location information. The hamming distance was a natural choice as a similarity measure between binary strings. More specifically, for each pixel in a patch, LBP was calculated as an 8-bit binary string and the similarity measure between two patches was the summation of the hamming distances at corresponding pixel locations. The reason we chose to transform the original image pixel values to binary strings using LBP was that in smooth regions, camera sensor noise would likely dominate the pixel-value variations within an image patch, which could be well captured by LBP encoding, because LBP

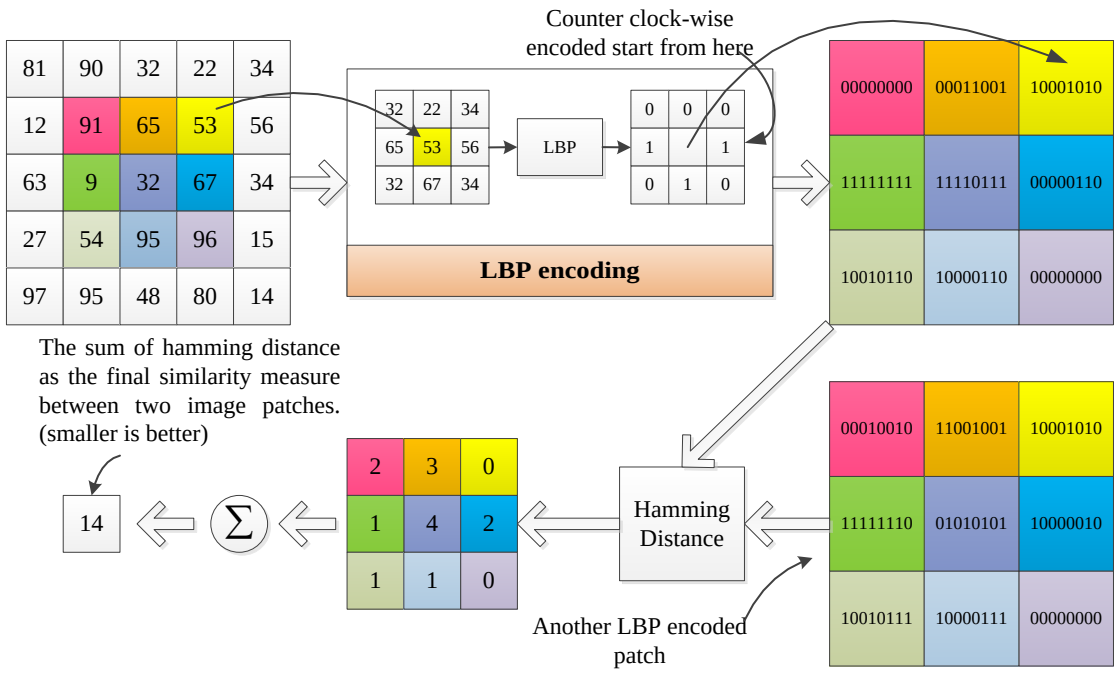


Figure 4.8 An example of coding pixel values in an image patch to bit strings, and calculating the summation of hamming distance as the similarity measure with another LBP-coded patch.

encoding considers the relative relationships of 8-neighbors with respect to the central pixel. If the copy-moved area is texture or object, LBP should also work. Therefore, the hamming distance of LBP encoded patches should work no matter the copy-moved regions were in smooth areas, texture areas, or were objects, as long as there was no further processing as mentioned at the beginning of Section 4.3. Figure 4.8 shows an example to generate LBP binary strings for one patch, the hamming distance calculation with another LBP-encoded patch, and the summation of the hamming distances as the similarity measure. Besides the sensitivity to post-processing, another drawback we discovered was that the proposed LBP-based similarity measure created lots of false positives in images with periodic patterns, such as images decoded from JPEG which have 8×8 block artifacts. Some examples of the results on the competition dataset are

provided in Figure 4.9 – 4.12. All the detection output with patch size of 5×5, 7×7, 9×9 and 11×11 are displayed, and the combined output was generated with the following post-processing steps:

1. Remove connected components with small areas.
2. Perform dilation on all the rest of the connected components to inrease the chance that the output mask covers all of the tampered region.
3. Combined the 5x5, 7x7, 9x9, 11x11 output masks together with pixel-wise OR.

Note that a lot of descriptors and distance measures have been proposed in the literature [93-101]; it would be our future work to evaluate their performance on the competition dataset.

4.3.2 A Simple Usage of SIFT

In this Section, we discuss how we use SIFT [15] to solve the problem when the copy-moved region has been further processed. We realize that there are a few publications that have addressed this problem using SIFT. It would again be our future work to evaluate their performance on the competition dataset. For this competition, a simple and somewhat naïve usage of SIFT was used.

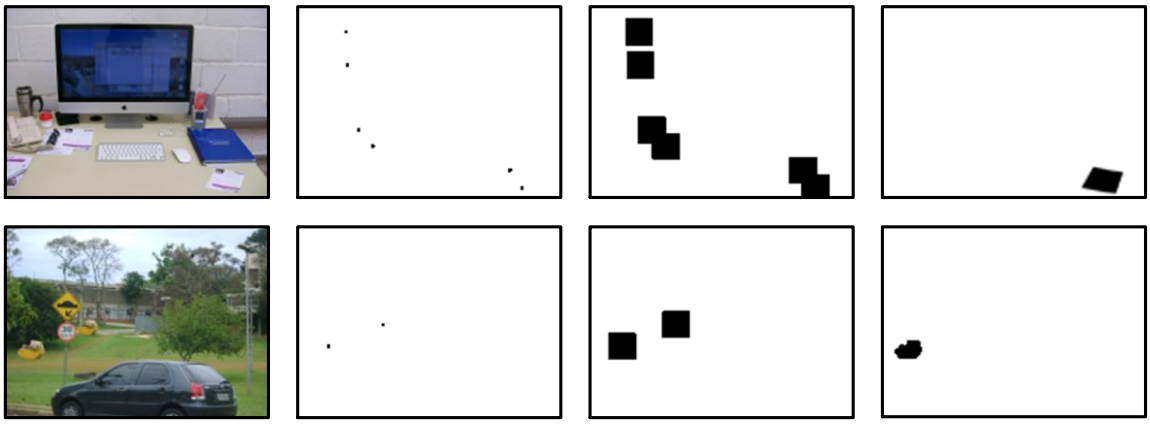


Figure 4.13 Two examples of successful tampering localization with the SIFT-based copy-move detector. From left to right: the original images, matched SIFT points, output after post-processing, ground truths.

The implementation of SIFT[21] was adopted as feature extraction tool. In the SIFT algorithm, each detected feature point is coded into a 128-D feature vector. Once all of the SIFT vectors have been generated for a test image, a brute force search was performed to find the matched points by the Euclidean distance between feature vectors. Then, all of the distances were compared with a pre-set threshold to locate the copy-moved regions. After that, morphological dilations were performed on every detected feature points to expend them to regions. Two examples of successful detections are given in Figure 4.13. This usage is quite coarse with still acceptable performance. Finally, the output of the SIFT-based detector will be combined with the LBP-based detector by pixel-wisely applying the logical OR operator.

[21] http://www.robots.ox.ac.uk/~vedaldi/code/sift.html

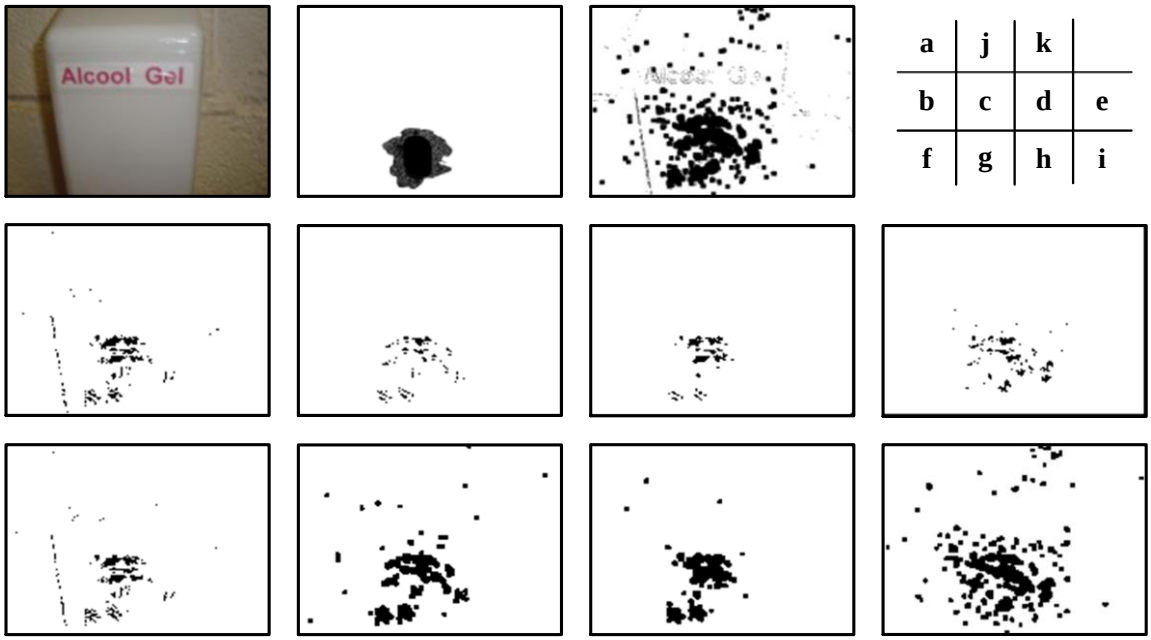


Figure 4.9 The first example of copy-move forgery in smooth regions. (a) The original image; (b) – (e) detected masks with patch size 5×5, 7×7, 9×9, and 11×11 respectively; (f) – (i) corresponding masks after post-processing; (j) the ground truth mask; (k) the final fused binary output mask.

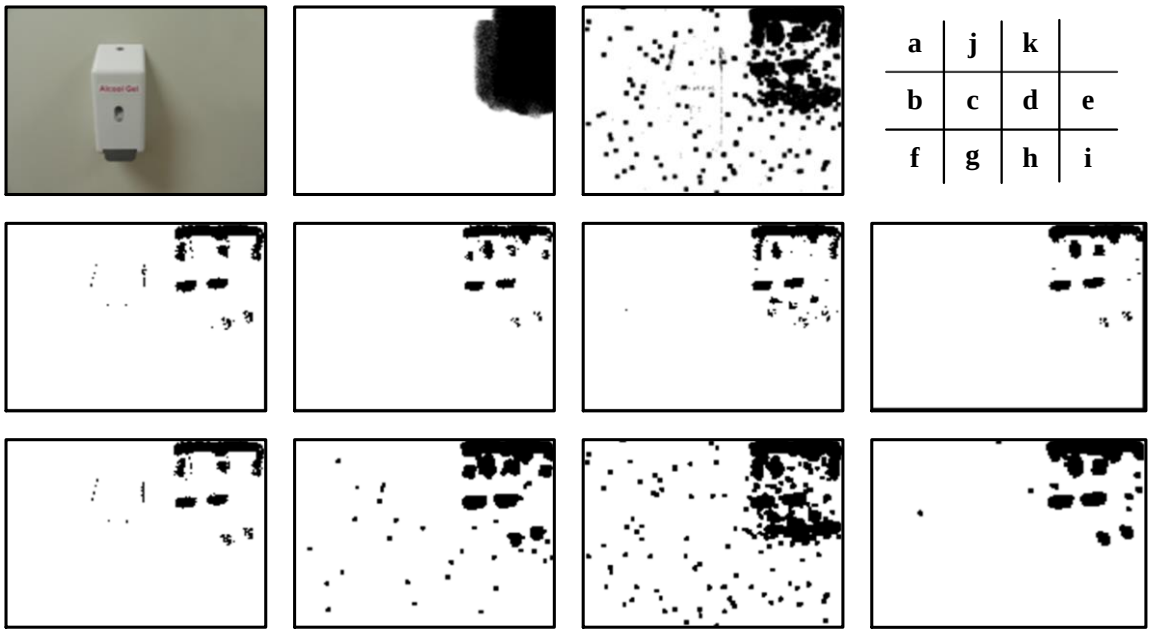


Figure 4.10 The second example of copy-move forgery in smooth regions. (a) The original image; (b) – (e) detected masks with patch size 5×5, 7×7, 9×9, and 11×11 respectively; (f) – (i) corresponding masks after post-processing; (j) the ground truth mask; (k) the final fused binary output mask.

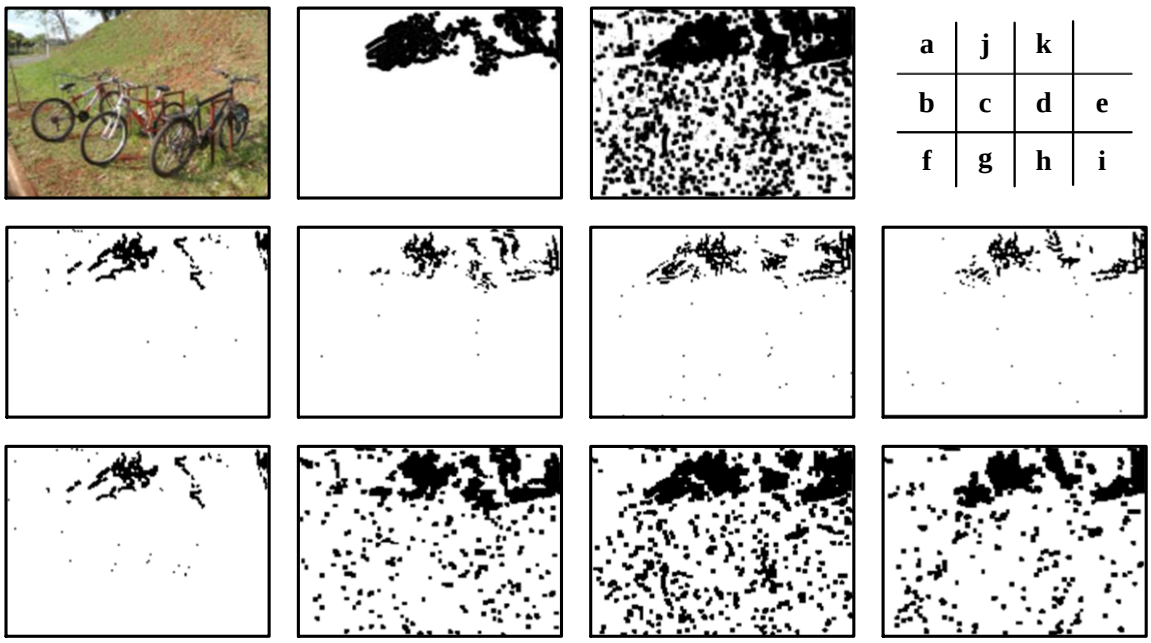


Figure 4.11 The first example of copy-move forgery in textural regions. (a) The original image; (b) – (e) detected masks with patch size 5×5, 7×7, 9×9, and 11×11 respectively; (f) – (i) corresponding masks after post-processing; (j) the ground truth mask; (k) the final fused binary output mask.

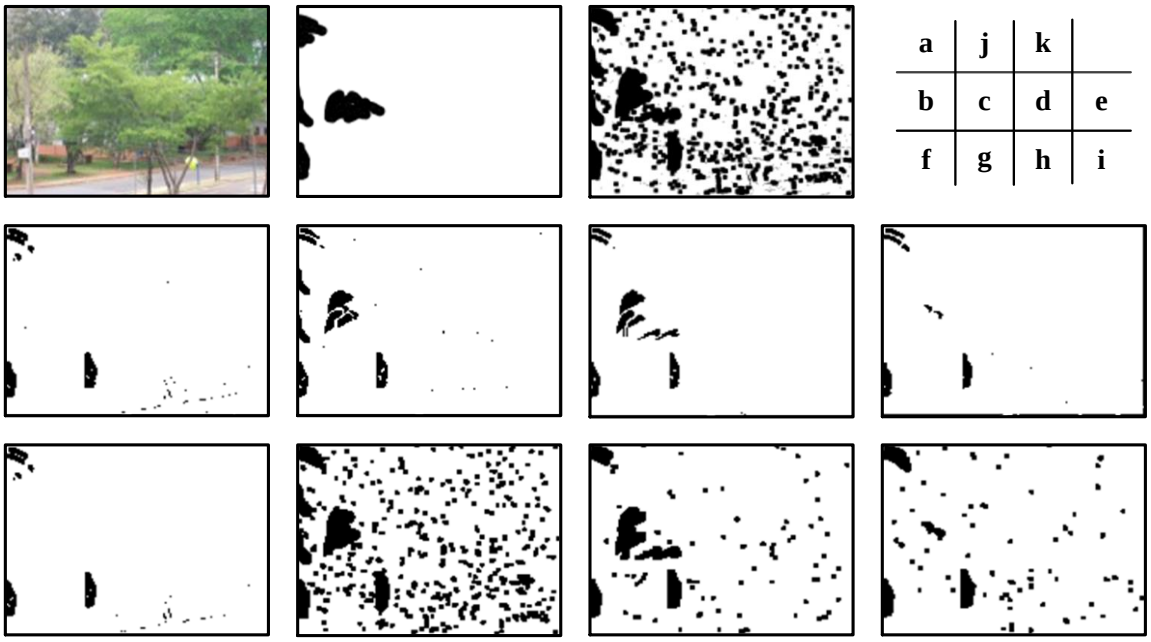


Figure 4.12 The second example of copy-move forgery in textural regions. (a) The original image; (b) – (e) detected masks with patch size 5×5, 7×7, 9×9, and 11×11 respectively; (f) – (i) corresponding masks after post-processing; (j) the ground truth mask; (k) the final fused binary output mask.

4.4 Methodology Comparisons with the Winner

In Phase-1 of the competition, the winner team [117] took the strategy of merging two tampering detection methods, a statistical feature-based classifier and a copy-move detector. Similar to our approach, they also used a subset of the rich model as the features. While we used ensemble classifiers and simultaneously considered classification performance and standard deviations of features and came up with an efficient subset of the original rich model, they used the SVM as classifier and the area under the receiver operating curve as the measure for feature selection. They also discovered that tampering detection with the feature-based method generated a lot of missed detections. This problem was alleviated by introducing a copy-move detector which by their experiments could efficiently 'catch' the missed detections. Eventually, the two tampering detection methods were merged and their testing score in phase-1 boosted to 94.2%.

In Phase-2 of the competition, the winner team [118] adopted and fused three methods, i.e., PRNU-based (photo response non-uniformity) tampering detection, copy-move detection, and statistical feature-based classification. Comparing with the other two, the PRNU-based detector was the most reliable one, when information of the camera noise was used. Although the camera information and noise patterns were not provided by the organizers, they were successfully uncovered by the winner team using a camera noise clustering methods on the training data. However, there were some camera mismatch between training and testing dataset, and the PRNU-based methods became unreliable at dark, saturated or highly textured regions. Hence, they adopted a copy-move detector as the second approach, which, surprisingly, also involved PatchMatch. The original version of PatchMatch was used which includes capability of detecting rotated

and scaled copy-moved regions. In their statistical feature-based method, the same feature set was used as developed during Phase-1 of the competition. Combined with a sliding window approach, the feature-based approach specifically targeted at splicing detection, although the reliability was deemed by the team as the lowest among the three methods.

In summary, the competition organized by the Technical Committee of Information Forensics and Security, IEEE Signal Processing Society has largely boosted the capability of image tampering detection by providing a large dataset and organizing the competition. The research on image tampering detection has thus been moved a big step. Many challenges, in particular how to identify the tampered regions, however, remain; and more advanced research is called for.

CHAPTER 5

SUMMARY

5.1 Major Contributions

In this dissertation, machine learning based (ML-based) methods have been developed to solve problems of image steganalysis and forensics.

In Chapter 2, in-depth studies have been conducted by the author to move the success achieved by the CNNs from computer vision to steganalysis. By analyzing the difference between steganalysis and computer vision, a CNN architecture incorporating knowledge of steganography and steganalysis, which is currently one of the best classifiers against advanced steganography, is proposed. This is the first work that outperforms traditional feature-based methods on using CNN for steganalysis. It also convinced the research society of CNN's capability and potential on steganalysis.

In Chapter 3, for camera model classification, two types of statistical features have been proposed to capture the traces left by in-camera image processing algorithms of different makes and models. The first type is Markov transition probabilities of the neighboring block-DCT coefficients for JPEG images, the second is based on histograms of local binary patterns (LBPs) obtained in both the spatial and wavelet domains of images. The designed feature sets serve as the input to support vector machines for classification. Both of the two feature sets achieve the top performance at the time they are proposed.

The last part of this dissertation documents the solutions delivered by the author's team to The First Image Forensics Challenge organized by the Information Forensics and

Security Technical Committee of the IEEE Signal Processing Society. In contrast to the common image tampering detection dataset created in a fully-controlled manner for pure research purposes, all the fake images involved in the challenge had been doctored by popular image-editing software to simulate the real-world scenario of tampering detection (images have been tampered or not) and localization (which pixels have been tampered); hence, the detection algorithms are required to be practical. The author's team won the runner-up prizes in both the two phases of the Challenge.

5.2 Discussion

The camera model classification addressed in this dissertation is one of the popular topics in image forensics and security. Besides its original function as the source identifier, it could also be applied to locate tampered regions in tampering detection, whenever the tampered region comes from images of different camera models. Moreover, it could also be served as pre-forensic steps to narrow down the range of candidate camera models when the investigators are looking for the specific camera device which has captured the image of interest. It can also be used to find suitable cover images and build a dataset which has closer statistical properties for more accurate steganalysis. The limitations of current feature-based camera model classification is the assumption that the testing images have not been post-processed, and they must all come from the camera models that have appeared in the training set.

5.3 Future Work

CNN-based steganalysis is expected to be future trend of steganalysis. Although a CNN architecture tailored for steganalysis is proposed, certainly, more sophisticated design is called for to further move the research ahead. Note that in this dissertation, we only work on steganography in the spatial domain. Since JPEG is the most popular image format, research efforts need to be devoted to design CNN against steganography in the JPEG domain. Similar to object segmentation in computer vision, in the future, research on locating embedding changes using CNNs is expected, and is also worth to be studied. Extension of the designed CNN in steganalysis to other research topics in forensics and security, e.g., tampering detection and source classification, can be another direction of future works.

The current feature-based camera model classification is not robust to image post-processing, such as resizing or recompression. This is an urgent issue that prevents camera model identification from real-life application. Therefore, more efforts should be devoted to solve this weakness. The other issue is that the trained classifiers would be guaranteed to deliver an error when the testing image originates from a camera model not included in the training set. One of the solutions is to train on all the existing camera models. It is unclear whether it is practical or not to take this approach. If not, more studies are demanded to overcome this weakness.

REFERENCES

[1] T. Filler and J. Fridrich, "Gibbs construction in steganography," *IEEE Trans. Inf. Forensics Security,* vol. 5, no. 4, pp. 705–720, Dec. 2010.

[2] T. Filler, J. Judas, and J. Fridrich, "Minimizing additive distortion in steganography using syndrome-trellis codes," *IEEE Trans. Inf. Forensics Security*, vol. 6, no. 3, pp. 920–935, Sept. 2011.

[3] V. Holub, J. Fridrich, and T. Denemark, "Universal distortion function for steganography in an arbitrary domain," *EURASIP J. Inf. Secur.*, vol. 2014, no. 1, pp. 1–13, Dec. 2014.

[4] L. Breiman, "Statistical modeling: the two cultures," *Statistical Sci.*, vol. 16, no. 3, pp. 199–215, 2001.

[5] B. E. Boser, I. Guyon, and V. Vapnik. "A training algorithm for optimal margin classifiers," in *Proc. Annu. ACM Workshop Comput. Learn. Theory (COLT)*, 1992, pp. 144–152.

[6] C. Cortes and V. Vapnik. "Support-vector network," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sept. 1995.

[7] H. Farid and L. Siwei, "Detecting hidden messages using higher-order statistics and support vector machines," in *Proc. Int. Workshop Inf. Hiding (IH)*, 2002, pp. 340–354.

[8] C. Chen, Y. Q. Shi, W. Chen and, G. Xuan, "Statistical moments based universal steganalysis using JPEG 2-D array and 2-D characteristics function," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, 2006, pp. 105–108.

[9] J. Fridrich, M. Goljan, and D. Hogea, "Steganalysis of JPEG images: Breaking the F5 algorithm," in *Proc. Int. Workshop Inf. Hiding (IH)*, 2002, pp. 310–323.

[10] J. Kodovský and J. Fridrich, "Calibration revisited," in *Proc. ACM Multimedia & Secur. Workshop (MMSec)*, 2009, pp. 63–74.

[11] D. Zou, Y. Q. Shi, W. Su, and G. Xuan, "Steganalysis based on Markov model of thresholded prediction-error image," in *Proc. IEEE Int. Conf. Multimedia and Expo. (ICME)*, 2006, pp. 1365–1368.

[12] J. Fridrich and J. Kodovský, "Rich models for steganalysis of digital images," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 3, pp. 868–882, June 2012.

[13] T. Pevny, P. Bas, and J. Fridrich, “Steganalysis by subtractive pixel adjacency matrix,” *IEEE Trans. Inf. Forensics Security*, vol. 5, no. 2, pp. 215–224, June 2010.

[14] V. Holub and J. Fridrich, “Random projections of residuals for digital image steganalysis,” *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 12, pp. 1996–2006, Dec. 2013.

[15] D.G. Lowe, “Object recognition from local scale-invariant features,” in *Proc. Int. Conf. Comput. Vis. (ICCV)*, 1999, pp. 1150–1157.

[16] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, “Speeded up robust features,” in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2006, pp. 404–417.

[17] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2005, pp. 886–893.

[18] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray, “Visual categorization with bags of keypoints,” in *Proc. Workshop Statistical Learn. Comput. Vis.*, 2004, pp. 1–22.

[19] S. Lazebnik, C. Schmid and J. Ponce, “Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2006, pp. 2169–2178.

[20] F. Perronnin, J. Sánchez and T. Mensink, “Improving the fisher kernel for large-scale image classification,” in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2010, pp. 143–156.

[21] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, “Locality-constrained linear coding for image classification,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2010, pp. 3360–3367.

[22] Y. Le Cun *et al.*, “Handwritten digit recognition with a back-propagation network,” in *Proc. Advances Neural Inform. Process. Syst. (NIPS)*, 1990, pp. 396–404.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Proc. Advances Neural Inform. Process. Syst. (NIPS)*, 2012, pp. 1106–1114.

[24] O. Russakovsky *et al.*, “ImageNet large scale visual recognition challenge,” *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[25] D. E. Rumelhart, G. E. Hinton; and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, pp. 533–536, 1986.

[26] S. Tan and B. Li, “Stacked convolutional auto-encoders for steganalysis of digital images,” in *Proc. Asia Pac. Signal Inf. Process. Assoc. (APSIPA) Annu. Summit Conf.*, 2014, pp. 1–4.

[27] T. Pevný, T. Filler, and P. Bas, “Using high-dimensional image models to perform highly undetectable steganography,” in *Proc. Int. Workshop Inf. Hiding (IH)*, 2010, pp. 161–177.

[28] P. Bas, T. Filler, and T. Pevný, “Break our steganographic system – the ins and outs of organizing BOSS,” in *Proc. Int. Workshop Inf. Hiding (IH)*, 2011, pp. 59–70.

[29] A. D. Ker and R. Böhme, “Revisiting weighted stego-image steganalysis,” in *Proc. SPIE, Electron. Imag. Secur. Forensics Steganogr Watermark. Multimedia Contents X*, 2008, pp. 5 1–5 17.

[30] J. Kodovský, J. Fridrich, and V. Holub, “Ensemble classifiers for steganalysis of digital media,” *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 2, pp. 432–444, Apr. 2012.

[31] Y. Qian, J. Dong, W. Wang and T. Tan, “Deep learning for steganalysis via convolutional neural networks,” in *Proc. SPIE Media Watermark. Secur. Forensics*, 2015, pp. 94090J–94090–J10.

[32] V. Holub and J. Fridrich, “Designing steganographic distortion using directional filters,” in *Proc. IEEE Int. Workshop Inf. Forensics Security (WIFS)*, 2012, pp. 234–239.

[33] Y. Jia, *et al*., “Caffe: Convolutional architecture for fast feature embedding,” in *Proc. ACM Int. Conf. Multimedia*, 2014, pp. 675–678.

[34] B. Li, M. Wang, J. Huang, and X. Li, “A new cost function for spatial image steganography,” in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, 2014, pp. 4206–4210.

[35] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 807–814.

[36] S. Ioffe and C. Szegedy, “Batch normalization: accelerating deep network training by reducing internal covariate shift,” in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2015, pp. 448–456.

[37] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proc. Artif. Intell. Statist. (AISTATS)*, 2010, pp. 249–256.

[38] Y. Q. Shi, P. Sutthiwan, and L. Chen, "Textural features for steganalysis," in *Proc. Int. Workshop Inf. Hiding (IH)*, 2012, pp. 63–77.

[39] L. Chen, Y. Q. Shi, and P. Sutthiwan, "Variable multi-dimensional co-occurrence histogram for steganalysis," in *Proc. Int. Workshop Forensics Watermark. (IWDW*), 2014, pp. 559–573.

[40] J. T. Springenberg et al., "Striving for simplicity: the all convolutional net," 2015, arXiv:1412.6806.

[41] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, Aug. 1996.

[42] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci*, vol. 55, no. 1, pp. 119–139, Aug. 1997.

[43] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Ann. Statist.*, vol. 29, no. 5, pp. 1189–1232, Oct. 2001.

[44] L. Breiman, "Random forests," *Mach. Learn*., vol. 45, no. 1, pp. 5–32, Oct. 2001.

[45] C. Szegedy *et al.*, "Going deeper with convolutions," In *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2015, pp. 1–9.

[46] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015, arXiv:1512.03385.

[47] J. Chen, X. Kang, Y. Liu, and Z. J. Wang, "Median filtering forensics based on convolutional neural networks," *IEEE Signal Process. Lett.*, vol. 22, no. 11, pp. 1849–1853, June 2015.

[48] T. Denemark, V. Sedighi, V. Holub, R. Cogranne, and J. Fridrich, "Selection-channel-aware rich model for steganalysis of digital images," In *Proc. 6th IEEE Int. Workshop Inf. Forensics Security (WIFS)*, 2014, pp. 48–53.

[49] W. Tang, H. Li, W. Luo, and J. Huang, "Adaptive steganalysis against WOW embedding algorithm," In *Proc. ACM Workshop Inf. Hiding Multimedia Secur. (IH&MMSec)*, 2014, pp. 91–96.

[50] G. Xu, H. Wu, and Y. Q. Shi, "Structural design of convolutional neural networks for steganalysis," *IEEE Signal Process. Lett.*, vol. 23, no. 5, pp. 708–712, Mar. 2016.

[51] K. S. Choi, E. Y. Lam, and K. K. Y. Wong, "Automatic source camera identification using the intrinsic lens radial distortion," *Opt. Express*, vol. 14, no. 24, pp. 11551–11565, Nov. 2006.

[52] T. Filler, J. Fridrich, and M. Goljan, "Using sensor pattern noise for camera model identification," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, 2008, pp. 1296–1299.

[53] B. E. Bayer, "Color imaging array," U.S. Patent 3 971 065, July 20, 1976.

[54] A. Swaminathan, W. Min, and K. J. R. Liu, "Nonintrusive component forensics of visual sensors using output images," *IEEE Trans. Inf. Forensics Security*, vol. 2, no. 1, pp. 91–106, Mar. 2007.

[55] Y. Long and Y. Huang, "Image based source camera identification using demosaicking," in *Proc. IEEE Workshop Multimedia Signal Process.*, 2006, pp. 419–424.

[56] S. Bayram, H. Sencar, N. Memon, and I. Avcibas, "Source camera identification based on CFA interpolation," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, 2005, pp. III–69–72.

[57] A. C. Popescu and H. Farid, "Exposing digital forgeries in color filter array interpolated images," *IEEE Trans. Signal Process.*, vol.53, no.10, pp. 3948–3959, Oct. 2005

[58] K. S. Choi, E. Y. Lam and K. K. Y. Wong, "Source camera identification by JPEG compression statistics for image forensics," *TENCON 2006 - 2006 IEEE Region 10 Conf.*, 2006, pp.1–4.

[59] K. L. Mehdi, H. T. Sencar, and N. Memon, "Blind source camera identification," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, 2004, pp. 709–712.

[60] T. Gloe, K. Borowka, and A. Winkler, "Feature-based camera model identification works in practice," in *Proc. Int. Workshop Inf. Hiding (IH)*, 2009, pp. 262–276.

[61] T. Gloe and R. Böhme, "The dresden image database for benchmarking digital image forensics," *J. Digital Forensic Practice*, vol. 3, no. 2-4, pp. 150–159, 2010.

[62] Y. Q. Shi, C. Chen, and W. Chen, "A Markov process based approach to rffective attacking JPEG steganography," in *Proc. Int. Workshop Inf. Hiding (IH)*, 2007, pp. 249–264.

[63] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971-987, July 2002.

[64] C. Tu, T. D. Tran, "Context-based entropy coding of block transform coefficients for image compression," *IEEE Trans. Image Process.*, vol. 11, no. 11, pp. 1271-1283, Nov. 2002.

[65] M. Weinberger, G. Seroussi, and G. Sapiro, "LOCOI: A low complexity context-based lossless image compression algorithm," in *Proc. IEEE Data Compress. Conf.*, 1996, pp. 140-149.

[66] C.-C. Chang, C.-J. Lin, "LIBSVM: a library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, Apr. 2011. Software available at http://www.csie.ntu.edu.tw/˜cjlin/libsvm.

[67] M. C. Stamm, M. Wu and K. J. R. Liu, "Information forensics: an overview of the first decade," *IEEE Access*, vol. 1, pp. 167–200, 2013.

[68] H. Farid, "Image forgery detection," *IEEE Signal Process. Mag.*, vol. 26, no. 2, pp. 16–25, Mar. 2009.

[69] T.-T. Ng, J. Hsu, and S.-F. Chang. Columbia Image Splicing Detection Evaluation Dataset. [Online]. Available: http://www.ee. columbia.edu/ln/dvmm/downloads/AuthSplicedDataSet/dlform.html (accessed on November 30, 2016)

[70] Z. Fan and R. de Queiroz, "Identification of bitmap compression history: JPEG detection and quantizer estimation," *IEEE Trans. Image Process.*, vol. 12, no. 2, pp. 230–235, Feb. 2003.

[71] W. S. Lin, S. K. Tjoa, H. V. Zhao, and K. J. Ray Liu, "Digital image source coder forensics via intrinsic fingerprints," *IEEE Trans. Inf. Forensics Security*, vol. 4, no. 3, pp. 460–475, Sept. 2009.

[72] A. C. Popescu and H. Farid, "Statistical tools for digital forensics," in *Proc. Int. Workshop Inf. Hiding (IH)*, 2004, pp. 128–147.

[73] F. Huang, J. Huang and Y. Q. Shi, "Detecting double JPEG compression with the same quantization matrix," *IEEE Trans. Inf. Forensics Security*, vol. 5, no. 4, pp. 848–856, Dec. 2010.

[74] Y. L. Chen and C. T. Hsu, "Detecting recompression of JPEG images via periodicity analysis of compression artifacts for tampering detection," *IEEE Trans. Inf. Forensics Security*, vol. 6, no. 2, pp. 396–406, June 2011.

[75] T. Bianchi and A. Piva, "Detection of nonaligned double JPEG compression based on integer periodicity maps," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 2, pp. 842–848, Apr. 2012.

[76] T. Bianchi and A. Piva, "Image forgery localization via block-grained analysis of JPEG artifacts," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 3, pp. 1003–1017, June 2012.

[77] H. Farid, "Exposing digital forgeries from JPEG ghosts," *IEEE Trans. Inf. Forensics Security*, vol. 4, no. 1, pp. 154–160, Mar. 2009.

[78] A. C. Gallagher and T. Chen, "Image authentication by detecting traces of demosaicing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2008, pp. 1–8.

[79] A. E. Dirik and N. Memon, "Image tamper detection based on demosaicing artifacts," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, 2009, pp. 1497–1500.

[80] A. Swaminathan, M. Wu and K. J. R. Liu, "Digital image forensics via intrinsic fingerprints," *IEEE Trans. Inf. Forensics Security*, vol. 3, no. 1, pp. 101–117, Mar. 2008.

[81] M. Chen, J. Fridrich, M. Goljan and J. Lukas, "Determining image origin and integrity using sensor noise," *IEEE Trans. Inf. Forensics Security*, vol. 3, no. 1, pp. 74–90, Mar. 2008.

[82] S. Bayram, I. Avcibas, B. Sankur, and N. Memon, "Image manipulation detection with binary similarity measures," in *Proc. 13th Eur. Signal Process. Conf.*, 2005, pp. 752–755.

[83] S. Bayram, I. Avcibas, B. Sankur, and N. Memon, "Image manipulation detection," *J. Electron. Imag.*, vol. 15, no. 4, pp. 041102-1-041102–17, Dec. 2006.

[84] W. Chen, Y. Q. Shi, and W. Su, "Image splicing detection using 2-D phase congruency and statistical moments of characteristic function," in *Proc. SPIE Secur., Steganogr. 9th Watermark. Multimedia Contents*, 2007, pp. 65050R–1–65050R–8.

[85] Y. Q. Shi, C. Chen, and W. Chen, "A natural image model approach to splicing detection," in *Proc. ACM Multimedia & Secur. Workshop (MMSec)*, 2007, pp. 51–62.

[86] C. Chen and Y. Q. Shi, "JPEG image steganalysis utilizing both intrablock and interblock correlations," in *Proc. Int. Symp. Circuits and Systems (ISCAS)*, 2008, pp. 3029–3032.

[87] G. Xu and Y. Q. Shi, “Camera model identification using local binary patterns.” in *Proc. IEEE Int. Conf. Multimedia and Expo. (ICME),* 2012, pp. 392–397.

[88] B. Zhang, Y. Gao, S. Zhao, and J. Liu, “Local derivative pattern versus local binary pattern: face recognition with high-order local pattern descriptor,” *IEEE Trans. Image Process.*, vol. 19, no. 2, pp. 533-544, Feb. 2010..

[89] C. Barnes, E. Shechtman, A. Finkelstein, and D. B Goldman, “PatchMatch: a randomized correspondence algorithm for structural image editing,” *ACM Trans. Graph.*, vol. 28, no. 3, pp. 24:1–24:11, July 2009.

[90] C. Barnes, E. Shechtman, D. B. Goldman, and A. Finkelstein, “The generalized patchmatch correspondence algorithm,” in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2010, pp. 29–43.

[91] V. Christlein, C. Riess, J. Jordan; C. Riess, and E. Angelopoulou, “An evaluation of popular copy-move forgery detection approaches,” *IEEE Trans. Inf. Forensics Security*, vol.7, no.6, pp.1841–1854, Dec. 2012

[92] A. Piva, "An overview on image forensics", *Proc. ISRN Signal Process.*, vol. 2013, pp. 1–22, 2013.

[93] X. Pan and S. Lyu, “Region duplication detection using image feature matching,” *IEEE Trans. Inf. Forensics Security*, vol. 5, no. 4, pp. 857–867, Dec. 2010.

[94] I. Amerini, L. Ballan, R. Caldelli, A. Del Bimbo, and G. Serra, “A SIFT-based forensic method for copy-move attack detection and transformation recovery,” *IEEE Trans. Inf. Forensics Security*, vol. 6, no. 3, pp. 1099–1110, Sept. 2011.

[95] J.-M. Guo, Y.-. Liu, and Z.-J. Wu, “Duplication forgery detection using improved DAISY descriptor,” *Expert Syst. Appl.*, vol. 40, no. 2, pp. 707–714, Feb. 2013.

[96] P. Kakar and N. Sudha, “Exposing post-processed copy-paste forgeries through transform-invariant features,” *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 3, pp. 1018–1028, June 2012.

[97] J. Fridrich, D. Soukal, and J. Lukas, “Detection of copy-move forgery in digital images,” in *Proc. Digi. Forensic Res. Workshop*, 2003.

[98] S.-J. Ryu, M. Kirchner, M.-J. Lee, and H.-K. Lee, “Rotation invariant localization of duplicated image regions based on Zernike moments,” *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 8, pp. 1355–1370, Aug. 2013.

[99] Q. Wu, S. Wang, and X. Zhang, “Log-polar based scheme for revealing duplicated regions in digital images,” *IEEE Signal Process. Lett.*, vol. 18, no. 10, pp. 559–652, 2011.

[100] S. Bravo-Solorio and A. K. Nandi, "Automated detection and localization of duplicated regions affected by reflection, rotation and scaling in image forensics," *Signal Process.*, vol. 91, no. 8, pp. 1759–1770, Aug. 2011.

[101] F. Y. Shih and J. K. Jackson, "Copy-cover image forgery detection in parallel processing," *Int. J. Pattern Recog. Artif. Intell.*, vol. 29, no. 8, pp. 1554004–1554017, Dec. 2015.

[102] X. Li, Y. Zhao, M. Liao, F. Y. Shih, and Y. Q. Shi, "Detection of tampered region for JPEG images by using mode-based first digit features," *EURASIP J. Adv. Signal Process.*, vol 2012, no. 1, pp. 190–199, Dec. 2012.

[103] J. Kodovský, J. Fridrich, "Steganalysis of JPEG images using rich models," in *Proc. SPIE, Electron. Imag. Secur. Forensics Steganogr Watermark. Multimedia Contents XIV*, 2012, PP. 83030A–83030A –13.

[104] M. Goljan, J. Fridrich, and R. Cogranne, "Rich model for steganalysis of color images," in *Proc. IEEE Int. Workshop Inf. Forensics Security (WIFS)*, 2014, pp. 185–190.

[105] M. Goljan and J. Fridrich, "CFA-aware Features for Steganalysis of Color Images," in *Proc. SPIE, Electron. Imag. Secur. Forensics Steganogr Watermark. Multimedia Contents XVII*, 2015, pp. 94090V–94090V–13.

[106] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.

[107] E. Chang, C. Shiufun, and D. Pan, "Color filter array recovery using a threshold-based variable number of gradients," in *Proc. SPIE, Sensors, Cameras, and Applications for Digital Photography*, vol. 3650, 1999, pp. 36–43.

[108] F. Marra, G. Poggi, C. Sansone, and L. Verdoliva, "A study of co-occurrence based local features for camera model identification," *Multimedia Tools and Applications*, vol. 75, no. 256, pp. 1–17, June 2016.

[109] A. Tuama, F. Comby, and M. Chaumont, "Camera model identification with the use of deep convolutional neural networks," in *Proc. IEEE Int. Workshop Inf. Forensics Security (WIFS)*, 2016, to appear.

[110] L. Baroffio, L. Bondi, P. Bestagini, and S. Tubaro, "Camera identification with deep convolutional networks," 2016, arXiv:1603.01068.

[111] T. Denemark, V. Sedighi, V. Holub, R. Cogranne and J. Fridrich, "Selection-channel-aware rich model for steganalysis of digital images," in *Proc. IEEE Int. Workshop Inf. Forensics Security (WIFS)*, 2014, pp. 48–53.

[112] W. Tang, H. Li, W. Luo, and J. Huang. "Adaptive steganalysis against WOW embedding algorithm," in *Proc. ACM Workshop Inf. Hiding Multimedia Secur. (IH&MMSec)*, 2014, pp. 91–96.

[113] W. Tang, H. Li, W. Luo and J. Huang, "Adaptive steganalysis based on rmbedding probabilities of pixels," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 4, pp. 734-745, Apr. 2016.

[114] T. Denemark, J. Fridrich and P. Comesaña-Alfaro, "Improving selection-channel-aware steganalysis features," in *Proc. SPIE, Electron. Imag. Secur. Forensics Steganogr Watermark. Multimedia Contents XV*, Feb. 2016, pp. 14–18.

[115] V. Holub and J. Fridrich, "Low-Complexity features for JPEG steganalysisu undecimated DCT," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 2, pp. 219–228, Feb. 2015.

[116] X. Song, F. Liu, C. Yang, X. Luo and Y. Zhang, "Steganalysis of adaptive JPEG steganography using 2D Gabor Filters," in *ACM Workshop Inf. Hiding Multimedia Secur. (IH&MMSec)*, 2015, pp. 15–23.

[117] D. Cozzolino, D. Gragnaniello and L. Verdoliva, "Image forgery detection through residual-based local descriptors and block-matching," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, 2014, pp. 5297–5301.

[118] D. Cozzolino, D. Gragnaniello and L. Verdoliva, "Image forgery localization through the fusion of camera-based, feature-based and pixel-based techniques," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, 2014, pp. 5302–5306.

[119] B. T. Polyak and A. B. Juditsky. "Acceleration of stochastic approximation by averaging," *SIAM J. Control Optim.*, vol. 30, no.2, pp. 838–855, July 1992.

[120] J. Long, E. Shelhamer and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, Boston, MA, 2015, pp. 3431–3440.

[121] L-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," 2016, arXiv:1606.00915.

[122] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proc. Int. Conf. Comput. Vis. (ICCV)*, 2015, pp. 1520–1528.

[123] J. Dai, K. He, and J. Sun, “BoxSup: exploiting bounding boxes to supervise convolutional networks for semantic segmentation,” in *Proc. Int. Conf. Comput. Vis. (ICCV)*, 2015, pp. 1635–1643.

[124] R. Girshick, J. Donahue, T. Darrell and J. Malik, “Region-based convolutional networks for accurate object detection and segmentation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 1, pp. 142–158, Jan. 2016.

[125] R. Girshick, J. Donahue, T. Darrell and J. Malik, “Region-based convolutional networks for accurate object detection and segmentation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 1, pp. 142–158, Jan. 2016.

[126] R. Girshick, J. Donahue, T. Darrell and J. Malik, “Region-based convolutional networks for accurate object detection and segmentation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 1, pp. 142–158, Jan. 2016.