# Machine Learning Techniques for Image Forensics in Adversarial Setting

**Ehsan Nowroozi**

Ph.D Dissertation in Information Engineering and Science
University of Siena

**UNIVERSITÀ DEGLI STUDI DI SIENA**

Facoltà di Ingegneria

Dipartimento di Ingegneria dell'Informazione e Scienze Matematiche

UNIVERSITÀ
DI SIENA
1240

# Machine Learning Techniques for Image Forensics in Adversarial Setting

Ehsan Nowroozi

*Ph.D Dissertation in Information Engineering and Science*
*XXXII Cycle, 2016-2019*

*Supervisor*
Prof. Mauro Barni

*Co-Supervisor*
Dr. Benedetta Tondi

*External reviewers*
Prof. Alessandro Piva
Prof. Giulia Boato

*Examination Committee*
Prof. Alessandro Piva
Prof. Giulia Boato
Prof. Stefano Melacci

Siena
APRIL 2, 2020

# Contents

# List of Figures

# List of Tables

# Acknowledgements

First and foremost I want to thank to my advisors, Prof. Mauro Barni, and my co-supervisor Dr. Benedetta Tondi, for supporting me during these past three years. Mauro is someone you will instantly love and never forget once you meet him. He's one of the smartest people I know. I wish that I could be as lively, energetic, and passionate as Mauro and to someday be able to give a command to the audience as well as he can. He has taught me how good experimental is done consciously or unconsciously. I appreciate all his contributions during my time and funding to make my Ph.D. experience fruitful. Thankful for the excellent example he has provided as an outstanding professor and researcher. Besides my advisors, I wish to thank my thesis reviewers and members of the committee, Prof. Alessandro Piva (University of Florence), Prof. Giulia Boato (University of Trento), and Prof. Stefano Melacci (University of Siena), for their insightful comments and suggestions. Furthermore, I want to acknowledge Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory (AFRL) for financial support during the Ph.D. course, and for making fruitful collaboration with professional researchers and laboratories.

I would like to thank all of my laboratory (VIPP) colleagues in Siena for their support and help: Kassem Kallas, and Bowen Zhang. A special thanks go to Benedetta Tondi and Andrea Costanzo who shared with me the intuition and the research work that led to many scientific publications.

Lastly, I would like to thank my mother (Ghezlarbas Ghaffari) for all her love and encouragement and who raised me with a love of science and

supported me in all my pursuits. Thanks to my father (Esmaeil Nowroozi - he was passed away on 1986) that reminds us love, respect, care, shelter, support, sacrifices, and many more. Besides, to my sisters, Nazanin, Negin and Noshin Nowroozi, and my brother Ebrahim Nowroozi, always encouraged me and supported me in all aspects during my studies.

*Ehsan Nowroozi*
*Siena, Italy*
*APRIL 2, 2020*

# Abstract

The *use of machine-learning for multimedia forensics is gaining more and more consensus, especially due to the amazing possibilities offered by modern machine learning techniques. By exploiting deep learning tools, new approaches have been proposed whose performance remarkably exceed those achieved by state-of-the-art methods based on standard machine-learning and model-based techniques. However, the inherent vulnerability and fragility of machine learning architectures pose new serious security threats, hindering the use of these tools in security-oriented applications, and, among them, multimedia forensics. The analysis of the security of machine learning-based techniques in the presence of an adversary attempting to impede the forensic analysis, and the development of new solutions capable to improve the security of such techniques is then of primary importance, and, recently, has marked the birth of a new discipline, named Adversarial Machine Learning.*

*By focusing on Image Forensics and image manipulation detection in particular, this thesis contributes to the above mission by developing novel techniques for enhancing the security of binary manipulation detectors based on machine learning in several adversarial scenarios. The validity of the proposed solutions has been assessed by considering several manipulation tasks, ranging from the detection of double compression and contrast adjustment, to the detection of geometric transformations and filtering operations.*

# Chapter 1

# Introduction

*"All objects in the universe are unique. No two things that happen by chance ever happen in exactly the same way. No two things are ever constructed or manufactured in exactly the same way. No two things wear in exactly the same way. No two things ever break in exactly the same way."*

Joe Nickell

*"We can all see, but can you observe?"*

A.D. Garrett, Everyone Lies

Nowadays, everybody has the possibility of editing the content of digital images and creating fake contents with relative easiness. While in many cases these are innocent operations, many other times image editing has malicious purpose. The diffusion of counterfeited image is a serious problem that impacts on judicial systems, global economy, financial health and homeland security [9, 10]. Multimedia Forensics was born as a new discipline aiming a gathering information on the history of a multimedia document, on its origin and authenticity, on the processing it underwent, etc... The use of Machine Learning (ML) for Multimedia Forensics, and Image Forensics, in particular, is gaining more and more consensus due to the powerfulness of ML tools and the possibilities offered by modern techniques based on deep learning architectures.

Disabling forensic analysis, however, turns out to be an easy task, given the weakness of the traces forensic techniques rely on. This is even more the case with ML-based methods relying on deep learning, due to the fragility of these algorithms and their inherent vulnerability to attacks. This makes it hard to exploit these tools in the a multimedia forensic scenario and more in general for security-oriented applications, where the possible presence of an adversary can not be ignored [11–14].

To address the security threats in Machine Learning, many researchers have started working on techniques for protecting learning systems, marking the birth of a new discipline, named Adversarial Machine Learning (Adv-ML). The concerns about the security of machine learning is exacerbated by the widespread of Deep Learning (DL) techniques. Deep Learning, in fact, suffers from a number of shortcomings hindering its application to security-oriented disciplines, and among them, Multimedia Forensics. The development of solutions capable to overcome the security limits of such technology in the presence of an adversary applying counter-forensics methods to impede the analysis, has therefore become a necessity in Multimedia Forensics.

This thesis contributes to the above mission with the development of machine learning techniques for Image Forensics in adversarial setting by focusing on image manipulation detection. In particular, we developed novel methods for increasing the security of binary manipulation detectors based on machine learning, including deep learning, in several adversarial scenarios.

The technical content of the thesis can be split in two parts, where two different approaches are considered for improving the security of forensic detectors. We first propose to secure standard ML-based forensic detectors by means of *adversarial training*, that is, by re-training the detector considering also 'properly crafted' attacked samples, in addition to the normal samples. The method is applied to the detection of double JPEG compression, which is one of the most studied problem in forensics. In this thesis, JPEG compression is regarded as a laundering-type attack, and JPEG-aware training is considered to design more robust detectors, by focusing on both Support Vector Machine (SVM) and Convolutional Neural Networks (CNN) classifiers. In the second part, a more general approach for designing an *intrisically more secure* classifier is adopted by working both on the architecture of the classifier and on the randomization of the feature set. In the case of feature selection randomization, we extended to CNNs an approach already considered in the forensic literature, and successfully applied to standard machine learning (SVM) classifiers [15].

## 1.1 Overview and Contribution

The thesis is organized as follows. Chapter 2 provides an introduction to Image Forensics and image manipulation detection, with a brief overview on the methodologies for forensic analysis. Then, we drive our focus on machine learning techniques developed for image forensic applications. Some basic concepts of Machine Learning (ML), necessary to understand the rest of the thesis, are also provided. A reader who is already familiar with the concepts of Image Forensics and ML tools can skip this chapter. In Chapter 3, we consider on adversarial setting and introduce the reader to Adversarial Multimedia Forensics by adopting the general view and terminology introduced in the field of Adversarial Machine Learning (AdvML). Specifically, in Chapter 3, the problem of Counter-Forensics (CF) is introduced and the related prior art is presented. Moreover, by adopting the perspective of the analyst, a classification of ML techniques that take into account the presence of the adversary is provided. The core of the thesis starts in Chapter 4. In this chapter, we design a general adversary-aware detector for the common problem of double JPEG detection, which is capable of detecting the double compression even in the presence of heterogeneous processing and CF attacks. Then, in Chapter 5, JPEG compression is regarded to as a laundering attack. By focusing on the detection of contrast enhancement, robustness to JPEG is achieved by building a JPEG-aware detector. We do so by considering both SVM and CNN architectures. The CNN architecture is proven to work under a wide variety of unseen tonal adjustments and when different software packages are used for compressing the images. In Chapter 6, we propose the use of an intrinsically more secure architecture based on multiple classifiers to improve the security of forensic SVM-based binary classification. The performance of the system are assessed, both in terms of robustness and security, for several image manipulation detection tasks and compared to traditional two-class classification architectures. Then, in Chapter 7, we analyze the security of the most recent CNN-based techniques developed for image forensics, by assessing the degree of *transferability* of the so-called adversarial examples under various settings, depending on the amount of knowledge available to the attacker about the target system. Then, In Chapter 8 we investigate on random feature selection approach which is proposed in [16] to improve the robustness of forensic detectors against targeted attacks, can be extended to detectors based on DL features. Finally, we conclude the thesis in Chapter 9, summarizing the lessons learned and outlining some ideas for future research in this direction.

## 1.2 Activity Within Research Projects

The US Department of Defense (DoD) would like to be able to extract knowledge from and understand this imagery and its provenance. Many images and videos are modified and/or manipulated prior to publication. The goal of this research is to develop a set of forensics tools to determine the integrity, semantic consistency and evolutionary history of images and videos. Moreover, the project gave me the opportunity to establish contacts with outstanding technical experts from seven universities, with complementary skills and background in computer vision and biometrics, machine learning, digital forensics, as well as signal processing and information theory[1].

The activity of this thesis has been partially supported by Defense Advanced Research Projects Agency (DARPA)[2] and Air Force Research laboratory (AFRL) under the research grant number FA8750-16-2-0173. The United States Government is certified to reproduce and distribute reprints for Governmental objectives notwithstanding any copyright notation thereon. The views and conclusions consist of herein are those of the authors and should not be explained as necessarily representing the official policies or authorization, either expressed or implied, DARPA and AFRL U.S. Government.

## 1.3 List of Publications

The research activity I carried out during my Ph.D. studies resulted in the following publications[3]:

[1] https://engineering.purdue.edu/MEDIFOR/

[2] https://www.darpa.mil/program/media-forensics

[3] *The list of authors is provided in alphabetic order.

*Chapter 4*
M. Barni, *E. Nowroozi*, and B. Tondi, "Higher-Order, Adversary-Aware, Double JPEG-Detection via Selected Training on Attacked Samples", In *25th European Signal Processing Conference (EUSIPCO)*, Kos, Greece, August 2017.

*Chapter 5*
M. Barni, *E. Nowroozi*, and B. Tondi, "Detection of Adaptive Histogram Equalization Robust Against JPEG Compression", In *International Workshop on Biometrics and Forensics (IWBF)*, Sassari, Italy, June 2018.

M. Barni, A. Costanzo, **E. Nowroozi**, and B. Tondi, "CNN-based detection of generic contrast adjustment with JPEG post-processing", In *25th IEEE International Conference on Image Processing (ICIP)*, Athens, Greece, October 2018.

*Chapter 6*
M. Barni, *E. Nowroozi*, and B. Tondi, "Improving the security of Image Manipulation Detection through One-and-a-half-class Multiple Classification", In *Multimedia Tools and Applications*, ISSN. 1573-7721, doi. 10.1007/s11042-019-08425-z, Springer, November 2019.

*Chapter 7*
M. Barni, K. Kallas, *E. Nowroozi*, and B. Tondi, "On the Transferability of Adversarial Examples Against CNN-Based Image Forensics", In *Multimedia Tools and Applications*, In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, United Kingdom, May 2019.

*Chapter 8*
M. Barni, *E. Nowroozi*, B. Tondi, B. Zhang " Effectiveness of random deep feature selection for securing image manipulation detectors against adversarial examples", Accepted paper in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Barcelona, Spain, *arXiv:1910.12392*, 4-8 May 2020.

## 1.4 List of Presentations

M. Barni, *$^*$***E. Nowroozi**, and B. Tondi, ”Higher-Order, Adversary-Aware, Double JPEG-Detection via Selected Training on Attacked Samples”, In *25th European Signal Processing Conference (EUSIPCO)*, Kos, Greece, August 2017.

M. Barni, $^*$**E. Nowroozi**, and B. Tondi, ”Detection of Adaptive Histogram Equalization Robust Against JPEG Compression”, In *International Workshop on Biometrics and Forensics (IWBF)*, Sassari, Italy, June 2018.

M. Barni, A. Costanzo, $^*$**E. Nowroozi**, and B. Tondi, ”CNN-based detection of generic contrast adjustment with JPEG post-processing”, In *25th IEEE International Conference on Image Processing (ICIP)*, Athens, Greece, October 2018.

Chapter 2

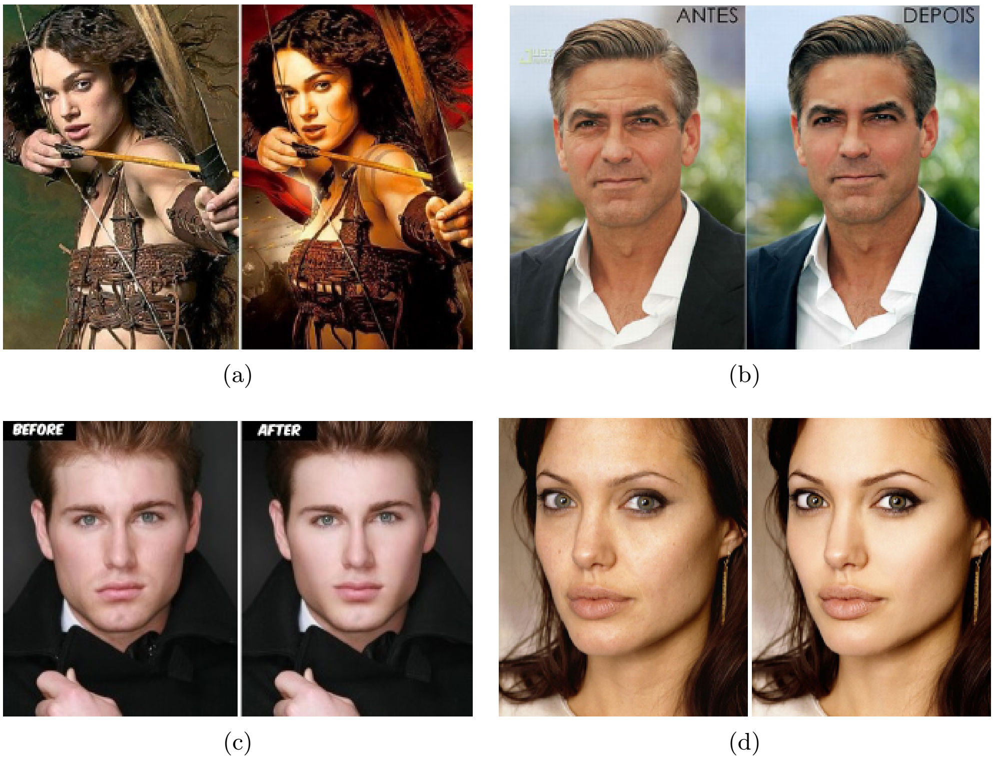
# Background on Image Forensics and Machine Learning

*"What one man can invent, another can discover."*
Sherlock Holmes

Over the past decades, digital photography is receiving a rapid and ever growing diffusion; it allows anyone to captures high-quality digital images, quickly and without cost, to easily store them on a large number of digital supports, and moreover share them on the Internet. At the same time, with the wide availability of image editing tools (e.g., Photoshop, Gimp,...), modifying on digital images has becomes extremely easy. Photo editing is frequently innocently used in entertainment, for instance, for improving the quality of images (see Figure 2.1). However, in many cases, photo editing is used malicious innocent intentions, as for instance in journalism (fake news), to modify the message conveyed by an image, or to conceal a subpart of it (see the examples in Figure 2.2). Tampering may also be used to falsify evidence in a court law.

*Digital Image Forensics* is a research field aiming at gathering information on the history of digital images in such a way that this trustability can be assessed [17, 18]. One of the strengths of Digital Image Forensics relies on its *blind* nature, which means that there is no access to the images before their diffusion: on *active* solutions alike Digital Watermarking [19] not viable, since they are based on concealing a message inside the image at creation time.

In the rest of this chapter, we first briefly introduce Image Forensics and present the main methods for forensic analysis. Then, we introduced some basic notions of Machine Learning and the most important techniques that have been exploited for forensic analysis, namely Support Vector Machines



(a) (b)

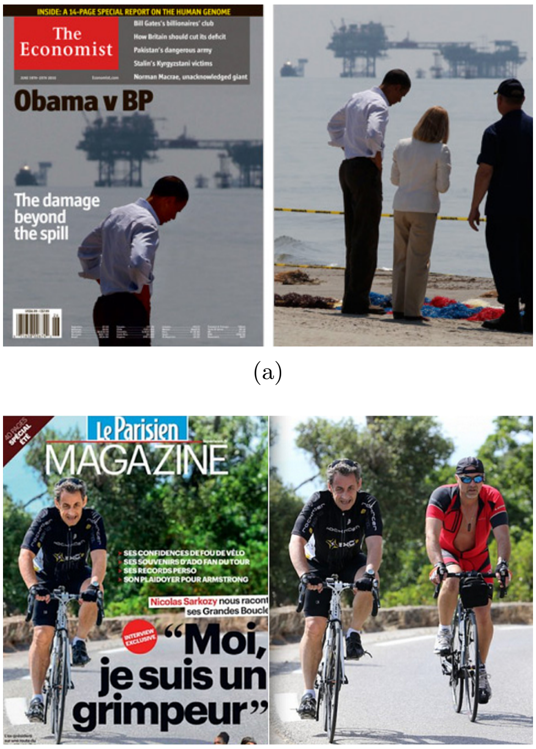
(c) (d)

Figure 2.1: Examples of image tampering before (left image) and after (right image) editing [1].

(SVMs) and Convolutional Neural Networks (CNNs). Finally, an overview of forensic techniques based on Machine Learning is provided.

## 2.1 What Is Image Forensics About?

The history of a digital image begins when the image is captured by the automatic application of some in-device processing. Other processing steps may follow such as the improvement of the perceptual quality of the image through enhancement operators, or altering the semantic content by including or removing objects. Several versions of the same image may be created, for


INSIDE: A 14-PAGE SPECIAL REPORT ON THE HUMAN GENOME
The Economist
Bill Gates's billionaires' club
How Britain should cut its deficit
Pakistan's dangerous army
Stalin's Kyrgyzstani victims
Norman Macrae, unacknowledged giant
Obama v BP
The damage beyond the spill
Le Parisien
MAGAZINE
SES CONFIDENCES DE FOU DE VÉLO
SES SOUVENIRS D'ADO FAN DU TOUR
SES RECORDS PERSO
SON PLAIDOYER POUR ARMSTRONG
Nicolas Sarkozy nous racont ses Grandes Boucl
INTERVIEW EXCLUSIVE
“Moi, je suis un grimpeur”

(a)

(b)

Figure 2.2: Examples of tampered images in journalism [2] (fake news).

instance by resizing it, or by applying compression, then modifying the file format. *Image Forensics* is about investigating the past history of the image, without knowing anything but the image itself.

Some of the main investigations, or forensic tasks, that have received great attention are reported in the following:

- *source classification*: whose objective is to determine whether the source image comes from a camera, a scanner, a cell phone, and so on.
- *source identification*: whose goal is to recognize the exact device which was used to capture the images.

- *Reverse engineering of processing operators*: which aims at detecting a chain of processing operators. Sometimes processing can be regarded to as a manipulation, then, this task is referred as *manipulation detection*. This is the category of forensic tasks considered in this thesis.

All the above investigations can be used for *Authenticity verification*, whose goal is to understand whether the image is original or has been manipulated a for malicious purposes.

## 2.2 Methods for Forensic Analysis

The basic observation Image Forensics relies on is the following: any processing carried out during any stage of an image's life cycle leaves specific subtle traces, namely *footprints*, that can be exploited by a forensic analyst to expose the corresponding processing or manipulation. The presence of these footprints can hence be investigated in order to gather information about the ”digital history” of the image.

The footprints that can be analyzed with forensics techniques are divided into three main categories [20]: *acquisition*, *coding*, and *processing* based footprints. Some examples of each category are discussed in the following without going into the details, a comprehensive overview of forensic techniques being not the object of this thesis.

### 2.2.1 Acquisition-based footprints

Every step of the acquisition process leaves some peculiar traces the within image. The captured image brings traces for instance of the particular Color Filter Array (CFA) pattern and the type of filter used for color interpolation [21]. As a result, the digital image brings evidence for both the employed CFA pattern and the interpolation filter [22–24]. Furthermore, a camera sensor, by default, leaves a particular noise in each captured image, which is known as Photo Response Non-Uniformity (PRNU). Such noise is unique for that specific sensor; hence, any of two different cameras leave the different patterns. The analysis of the traces described above permits to also detect other kinds of sources of digital images such as scanners and computer graphics [25, 26].

### 2.2.2 Coding-based footprints

Nowadays, most of the digital cameras use the JPEG standard compression format for efficient storage and transmission. Due to the widespread diffusion of the format, and since in many cases it turns out to be a useful asset for assessing the processing history of an image, a significant effort has been dedicated by the research community to the study of the compression history of an image [27–29]. Since different imaging softwares generally consider different compression parameters and uses different quantisation tables [27], the analysis of inconsistencies in quantization matrices can also be used for source and forgery detection [30]. Even more, by the analysis of coding-based (or compression-based) footprints, it is also possible to understand whether an image has been compressed multiple times. When an image is recompressed, the new 8× 8 JPEG grid is superimposed to the already existing one, either aligned or misaligned. In [31], the authors showed that, when the grid is aligned, double quantization entailed by DJPEG compression leaves peculiar artifacts in the histograms of DCT coefficients, especially at low and medium frequencies. When the second compression is weaker than the first one, the matrix with the first quantization steps can also be derived via statistical analysis, by exploiting the properties of integers numbers when they are quantized and rounded multiple times [32]. In [21], the authors derive a method based on a unified statistical model characterizing the DCT coefficients of the primary compression in the presence of both aligned and non-aligned DJPEG compression.

### 2.2.3 Editing-based footprints

These kinds of footprints are left within an image during image manipulation. Editing images is easy today thanks to the availability of powerful editing software (e.g., Photoshop, GIMP), but many subtle traces are left into the image during the manipulation.

In general, an editing tool can be used in a legitimate way, e.g., to enhance the quality and appearance of an image, or in an malicious way to alter the semantic content of the image. The above distinction is not always clear, as an harmless processing can be used with malicious intents; then, an editing operation can reveal or *be telltale of* a manipulation carried out for malicious purposes. For instance, filtering or resampling can hide traces of a previous tampering, or color adjustments can help to convey a different message with respect to that of the authentic image, thus changing the semantics of an image. For example, an image can be darkened to convey a sensation of menace or sadness.

The application of a geometric transformation, like a rotation or a resizing, requires interpolation of pixel values, that leaves detectable traces in the image [20, 33]. To provide an example, by applying a geometric transformation such as resizing and rotation, we are forcing the software to interpolate pixel values thus leaving detectable traces within the digital image [34]. Furthermore, by pasting a patch of pixels in a subpart of the image we introduce inconsistencies in terms of blurriness, contrast, and saturation of various areas of the same picture [35, 36].

Contrast enhancement is a common manipulation aiming of increasing the perceived quality of images. Usually, such enhancement is carried out by means of histogram equalisation, whereby the intensity values of pixels in the input image are remapped in such a way that the output image has a uniform distribution of intensities. Global contrast enhancement can be detected as explained in [37], that exploits the presence of artefacts (peaks and gaps) in the image histogram caused by an alteration of the contrast is exploited.

Median filtering has many applications in image processing including denoising and smoothing, and it can also be used to conceal traces of previous processing. Three main model-based algorithms detecting median filtering based on the following statistical properties have been proposed: the probability of two adjacent pixels being equal is larger for median filtered images [38]; the difference between two adjacent pixels is often zero [39]; the block-wise approach used by median filtering introduces correlation between blocks [40].

As stated in the Introduction of the thesis, our contribution is focused on authenticity verification, and more specifically on manipulation detection. Then, the main categories of footprints the developed tools rely on are compression-based and processing-based footprints.

## 2.3 Some Basic Machine Learning Concepts

In this section, we introduce the basic concepts of Machine Learning (ML), which are necessary to understand the techniques developed in this thesis.

Machine Learning is about learning distributions from input data. Machine Learning algorithms are mainly divided into two classes: *supervised* and *unsupervised* learning methods. Supervised learning is the most popular type of algorithms in practical applications, e.g. in pattern recognition, classification and image processing, due to its superior performance. Supervised learning algorithms build a model based on input 'labeled' data, that is, data for which the true class is known, sampled from the input distribution of data. The set of labeled data is referred to as *training data*.

When using ML to get a statistical model (for a classification task under analysis) from input data, there are many aspects that require attention. In particular, *overfitting* and *underfitting* are the deficiencies that a model's performance might suffer from.

- *Overfitting*: it occurs when a model has been over-trained on the input data. This may happen when too many features are considered for the input data space or because not enough data has been supplied. The models that have been overfitted on the training data do not generalise well to new examples, that is, they are not good at predicting unseen data.

- *Underfitting*: it is when the model has not learned enough from the training data, that is, it has not captured the underlying structure of the data, resulting in low generalization and unreliable predictions.

During the procedure of learning models and assessing their performance, the set of available data is divided into a *training set*, used for model training, and a *test set*, used for performance measuring. Training data is usually split into a training set and a validation set to assess the behavior with respect to unseen data and then optimize the choice of the internal parameters of the algorithm.

In the following, we describe two of the most relevant supervised ML algorithms, that have been adopted for the work of this thesis: Support Vector Machines (SVM) and Convolutional Neural Networks (CNNs).

### 2.3.1 Support vector machines

*Support Vector Machines (SVMs)* are supervised Machine Learning algorithms, that are widely used for classification tasks. Although they are designed for binary classification, multi-class classification also can be performed [41]. The objective of the support vector machine algorithm is to find a hyperplane in an $N$-dimensional space ($N$ is the number of features) that distinctly classifies the data points. To separate two classes of data points, many possible hyperplanes can be chosen. The goal is to find a hyperplane that has the maximum margin, i.e. a maximum distance between data from different classes. Actually, the maximum distance plays an important role in SVM, which provides a guarantee that future data points will be correctly classified with high confidence. The optimal hyperplane is depicted in 2.3 for a two-class case in two dimensions. In this figure, the data point (red and blue data points) falling on the two different sides of the hyperplanes are attributed to different classes. Moreover, the input data points that are near to the hyperplane, known as *support vectors* (SV), can impact on the position of the hyperplane. Hence, SVMs can maximize the margin of the classifier [15]. SVMs efficiently perform also non-linear classification by means of kernel trick, which works by mapping the input data into a higher-dimensional feature spaces, where the classification boundary is a hyperplane. Some popular kernels include the polynomial kernel, the Gaussian Radial Basis Function (RBF), the hyperbolic tangent.

The effectiveness of SVM depends on the selection of kernel, the kernel's parameters, and a soft margin parameter $C$. The parameter $C$ rules the tradeoff between the margin of the separating hyperplane in the higher dimensional space (the transformation of the input $x$ into the higher-dimensional space defines the kernel [42]) and the misclassification of the training points. The most common choice of the kernel is the Gaussian RBF, which has a single parameter $\gamma$, which determines the width of the kernel and then determines how far the influence of a training sample reaches. The best combination of $C$ and $\gamma$ is often selected by a grid search with exponentially growing sequences of $C$ and $\gamma$. Typically, each combination of parameter choices is checked using cross internal validation, and the parameters with best cross internal validation accuracy are picked (more details on this procedure are given in Chapter

6.

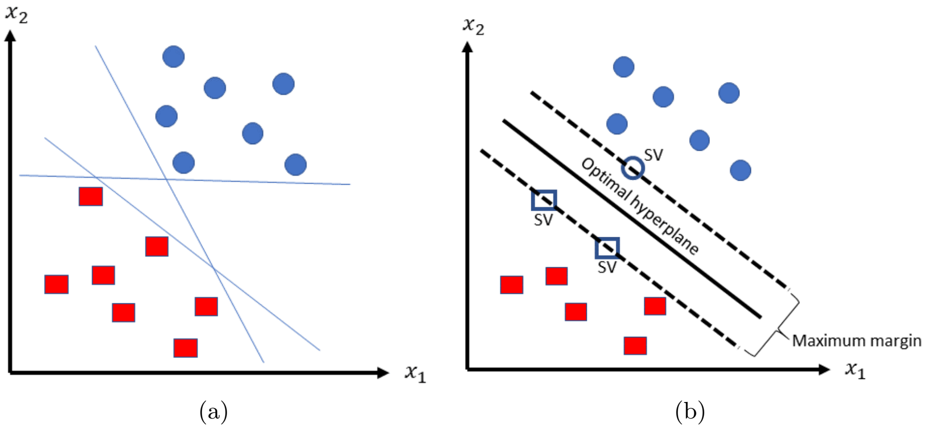


Figure 2.3: Illustration of the possible separating hyperplanes (a) and the SVM optimum hyperplane (b) in two dimensions (N=2).

### 2.3.2 Convolutional neural networks

Neural networks are a biologically-inspired programming model that allows a computer to learn from observational data. Deep learning networks are distinguished from standard single-hidden layers neural networks by their depth. Convolutional Neural Networks (CNNs) is a class of deep neural networks (DNNs) that can efficiently address several tasks in image and pattern recognition, image processing, and other close areas [43], with amazingly good performance. CNN is a complex computational model that consists of a large number of interconnected neurons. A weight parameter and a bias is associated to every neuron.

The set of operations in a CNN typically comprises convolution, non-linear activation and thresholding, and local pooling. By minimizing a cost function at the output of the last layer, the network parameters are tuned so that they are able to capture patterns in the input data and automatically extract distinctive features. In this way, the networks are able to learn complex functions of the input. In image processing applications, the CNN is directly

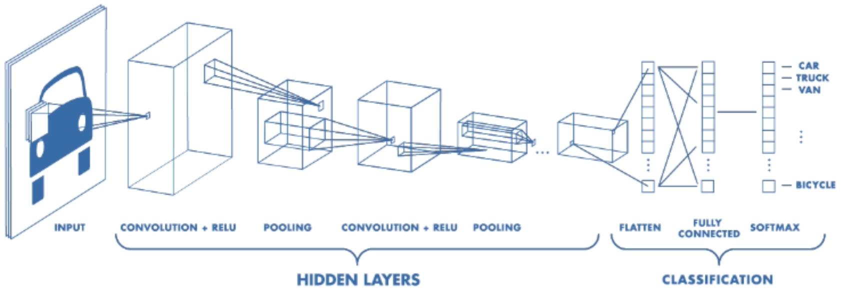


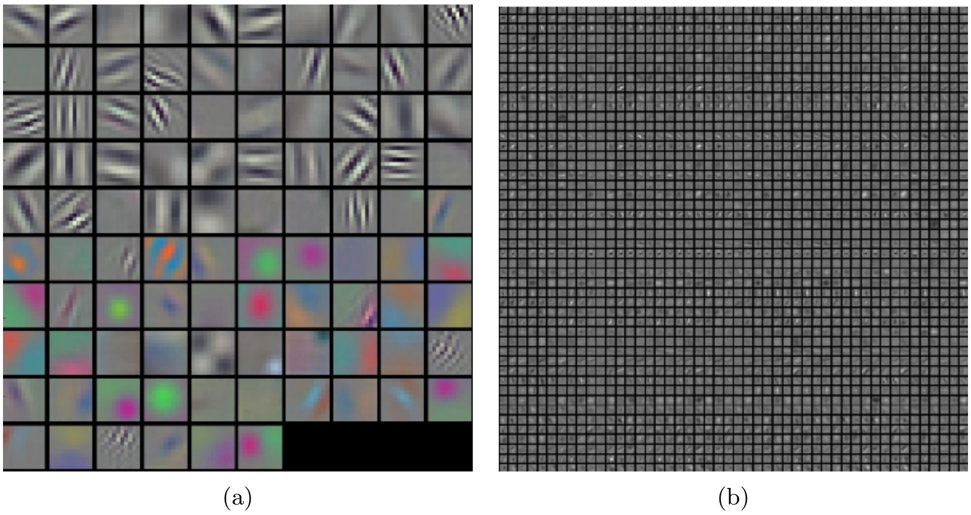
Figure 2.4: A simple CNN architecture [3].

fed with the input image. Therefore, in a CNN, the feature extraction process is completely driven by data, whereas in the traditional ML approaches, the process is driven by human intuition, through the selection of handcrafted features.

The main steps to build a CNN for image classification are: i) definition of the architecture, that is, the layers, the shape of the filters, etc; ii) definition of the loss function which is minimized during the training; iii) definition of a large dataset for training and testing with specific labels.

Figure 2.4 shows a simple example of CNN architecture. A CNN consists of two main parts: a convolutional part (hidden convolutional layers) and a fully connected part. The convolutional part performs feature extraction, while the fully connected part is used for classification. some of the most common layers are the following:

- *Convolution layer*: each convolution layer consists of a group of filters. Given an input, the output of each filter is obtained by applying a linear convolution with kernel size $k \times k$, with stride $S$, and is called feature map (see Figure 2.5). The strides defines the number of pixels shifts over the input matrix. For instance, when $S$ is two, then we move the filters two pixels at a time. Hence, the output of a convolutional layer consists of feature maps that are obtained by convolving the input with different filters.



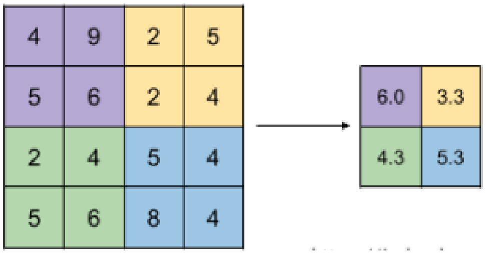
(a)

(b)

Figure 2.5: Example of typical-looking filters on the first convolutional layer (a) and second convolutional layer (b) of a trained AlexNet [4].

- *Pooling layer*: this layer performs down-sampling of the input. A pooling layer is usually added after a convolutional layer and is used to reduce the dimensionality of the feature maps. Several pooling operations (like a filter) can be applied with a given size (usually $2 \times 2$) to feature maps. The two most common functions used are: average pooling (or avg-pooling), that calculate the average value of the feature map over the pooling window; maximum pooling (or max-pooling), that consider the maximum value. The two pooling methods summarize the average presence of a feature (avg-pooling) and the strongest (most activated) presence of a feature (max-pooling). An example of the average pooling layer is provided in Figure 2.6.

- *RELU*: this layer performs element-wise nonlinear activation by applying to the input $x$ the rectification function $\max(0, x)$, i.e., truncating the negative values to zero, thus causing a nonlinear behavior. Other activation functions can be considered, such as sigmoids and hyperbolic tangents.



| | | | |
|---|---|---|---|
| 4 | 9 | 2 | 5 |
| 5 | 6 | 2 | 4 |
| 2 | 4 | 5 | 4 |
| 5 | 6 | 8 | 4 |

| | |
|---|---|
| 6.0 | 3.3 |
| 4.3 | 5.3 |

Figure 2.6: Example of average-pooling layer, with size 2 × 2, and stride $S = 2$ [5].

- *Fully-Connected layer*: this layer, also called inner-product layer, performs dot product between the input feature vector and a weight matrix.

- *Softmax layer*: this layer normalizes an input feature vector to a vector with the same number of elements summing to one. Often applied in the last layer of the network to transform the output soft values $y$ (logits) in probability values $p$ (softmax scores) which is expressed by,

$$S(y_i) = \frac{e^{y_i}}{\sum\limits_j e^{y_j}}, \tag{2.1}$$

During training, the weights of the convolutional and the fully-connected layers are learned. CNN models are trained by resorting to backpropagation [44], by means of optimization methods like gradient descent [45]. A loss functions is defined between the predicted and the expected output. The first layers of the networks typically learn low-level visual objects such as edges, simple shapes and color contrast, whereas deeper layers combine such information from previous layers to identify more complex visual patterns. In the last layer, the data are combined based on the cost function so that the loss is minimized. The set of training samples is usually split into small batches, sometimes called mini-batches, that is a small number of samples from all the classes (a small portion of the entire training set) [46]. For each batch the predictions for each sample are first computed (forward pass), then the gradients with respect to the loss function are averaged to update the filters' weights (backward pass). This forth and back process on a batch is called iteration. Furthermore, each time a different mini-batch is considered. When the entire training dataset is visited based on mini-batches, an epoch ends.

As we said before for general ML, training data are split into a training set and a validation set. In the case of CNNs, the training set is used to drive the gradient descent method to update the filters's weights by means of backpropagation, while the validation set is used to compute the loss over a set of samples unknown to the training process, so to estimate the behavior on unseen data.

## 2.4 Prior Art on ML-Based Image Forensics

As we have seen in Section 2.2, dedicated features have been developed in forensic literature to reveal the presence of many kinds of footprints in images. However, especially when more complex forensic tasks are addressed, a statistical characterization and modeling for the studied problems is often not available. In these cases, forensic researchers have resorted to ML techniques to address forensic tasks. ML tools are in fact capable to learn complicated patterns from a set of *hand-crafted* features, which cannot be revealed by standard model-based tools for statistical analysis. These patterns can then be used for the classification (looking for their presence or absence). With more complex ML and modern Deep Learning (DL) techniques, it is also possible to extract discriminative features directly from the input image. In this case, then, the features are totally *self-learned* (or automatically learned) from the input data.

In the following, we provide an overview of ML-based approaches developed for forensic tasks.

### 2.4.1 SVM-based image forensics

Many state-of-the-art Machine Learning-based Image Forensic methods rely on SVM classification, due to its simplicity and the good accuracy results that can be achieved for many classification tasks. In most of the early forensic methods, the features were hand-crafted, extracted from the image based on some heuristics, and very specific for the problem at hand. This is the case, for instance, of forensic methods developed for Double JPEG (DJPEG) compression or re-compression detection, which is one of the most widely studied problems in Image Forensics. Inspired by [31], the most popular approach for detecting double aligned JPEG (A-DJPEG) compression consists in analyzing the histogram of block-DCT coefficients. In [47], the authors proposed a detector based on SVM classifiers with feature vectors formed by histograms of low-frequency DCT coefficients. Other methods based on features extracted from the histograms of DCT coefficients and SVM are proposed in [48, 49]. In another work [50], a set of features designed to enhance DJPEG artifacts are considered to distinguish between double and single JPEG images, and used to feed an SVM-based classifier. Blocking artifacts in the pixel domain are investigated in [51], and their periodic property is measured by devising proper sets of features fed to a SVM; this method can work for non-aligned double JPEG (NA-DJPEG) detection. The authors in [29] considered the traces left by DJPEG in the mean, variance and entropy of the image, by training the SVM classifier based on all these statistical features together. More recently in [52], the first significant digits (FSD) of the DCT coefficients, namely, a statistic derived from the DCT histograms, have been used to tell apart single compressed images from double compressed ones. By exploiting the fact that the distribution of the FSDs of the DCT coefficients in single compressed images follows a generalized Benford's law, a model-based approach was first adopted in [53] for this purpose. Later, such features have been used in conjunction with SVM classification to distinguish between single from multiple compressed images and estimate the number of compression stages [52]. In [48], localization of spliced regions is achieved by using FSD features of block-DCT coefficients and employing a SVM classifier.

Other examples can be made for other forensic tasks. For instance, for the detection of global contrast enhancement, we mention the work in [54], where the authors considered statistics derived from the distribution of block variance and AC-DCT coefficients to feed an SVM classifier. The authors in [55] proposed a method for splicing and copy-move detection that exploits SVM classification of Local Binary Patterns (LBP) descriptors extracted from the block DCT of chroma channels. SVMs have been often adopted also for camera model identification, to perform multi-class classification based on high order features extracted from the images [56], ad-hoc features obtained from aberration measurements [57], or features extracted from the Photo Response Non Uniformity (PRNU) noise residual [58].

#### 2.4.1.1 Rich feature models and SVM classification

In early forensic methods based on statistical analysis, dedicated hand-crafted features were first determined, capable to reveal the presence of particular footprints and then expose the manipulation the forensic analysis is looking for, eventually resorting to data-driven or ML tools (e.g. SVMs) for classification.

In the last decade, there was a trend toward the use of general high-dimensional feature sets, that can provide rich image representations, so called *rich feature models*: the Spatial Rich Models (SRM) [59], and the Color Spatial Rich Models (CRM) [60], extending the previous set to the case of color images. The concept of rich features was introduced for steganalysis, then the use of rich feature sets has been extended to forensic tasks. Through the use of this set of features, based on co-occurrences of various noise residual images (obtained by applying high-pass linear and non-linear filters), various types of relationships among neighboring samples can be captured. In many cases, the adoption of rich feature sets permitted to boost the performance of state-of-the-art forensic classifiers based on ad-hoc features.

Rich features sets have been recently used for camera model identification [61–63], for the detection and localization of tampering [64], contrast-enhancement detection [65], median filtering detection [38], and, more in general, for the detection of several types of image processing operations [66]. Given the large dimensionality of the feature sets, these techniques often require to resort to more complex Machine Learning techniques, e.g. Random Forest or *Ensamble* classifiers. In many cases, a subset of these rich features is identified and used, e.g. in [61, 63], or the initial set of features is reduced (e.g. via Principal Component Analysis), as in [62], so that it is possible to resort again to SVM classification. Another commonly adopted strategy to achieve this purpose is to reduce the feature dimensionality of the original rich model by considering a smaller value for the truncation of the quantized image residuals and low order co-occurrences, as done in [66], where an SVM is used to classify several image processing operations based on rich features. Rich feature models borrowed from steganalysis coupled with SVMs have been also used for splicing detection [67] and forgery detection [68].

#### 2.4.1.2 A brief Introduction to Rich Image Representations and the SPAM feature set

We focus on the description of the rich feature sets in the pixel (spatial) domain, which are the most used especially in forensics (see Section 2.4.1.1).

These features are based on co-occurrence matrices computed on the thresholded prediction-error image, also called 'residual image'. The underlying idea is the following: modeling the residuals rather than the pixel values is often better for steganalysis and forensic applications, since the image content does not help detecting local alterations and should be suppressed altogether. As a further advantage, the residual image has a much narrower dynamic range than the original one, allowing for a compact and robust statistical description by means of co-occurrences. The analysis outlined above can be summarized in the following steps:

- computation of high-pass residuals;
- truncation and quantization;
- feature extraction based on co-occurrence matrices (of a given order);

Considering different high-pass filters, different image residuals can be obtained and then different feature sets. In [59], a large number of models have been obtained by considering many different high-pass filters, both linear and nonlinear, with various supports, different quantization and truncation factors for the residues. Specifically, 39 different high pass filters are proposed which work on the grayscale version of the original image $I$. The simpler one is the first order (horizontal) linear filter (from left to right), which produces the residues $r_{ij} = I_{i,j+1} - I_{i,j}$. The residual noises in all the directions (horizontal, vertical, diagonal) and orientations (left to right and right to left, up to down and down to up) are computed $\{\leftarrow, \rightarrow, \uparrow, \downarrow, \nwarrow, \searrow, \swarrow, \nearrow\}$ and then combined by exploiting the symmetry of natural images. Figure 2.7 shows the effect of applying this filter to an image.

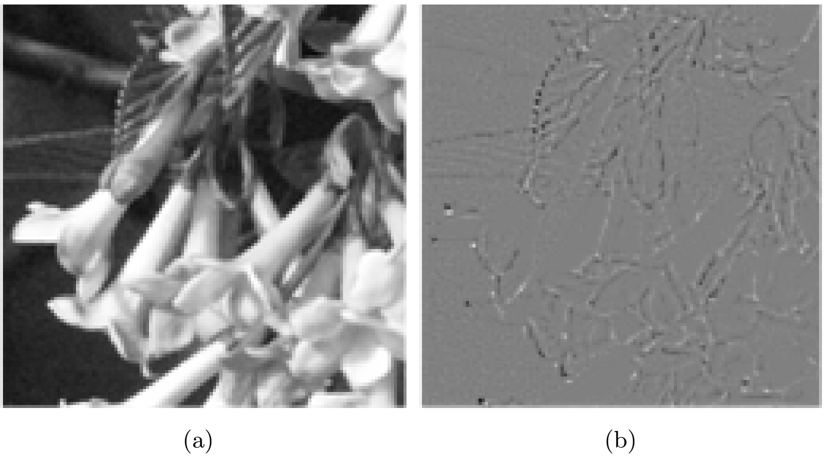


Figure 2.7: An image (a), and the corresponding residual image (b).

Residuals are real-valued and, although typically small, they span wide range. To enable meaningful characterization in terms of co-occurrence they must be quantized and truncated. The truncated residual is then obtained as

$$r_{ij} = \mathrm{trunc}_T(\mathrm{round}(r_{ij}/q)) \tag{2.2}$$

where $q$ is the quantization step and $T$ the truncation value. Each quantized residual can then take $2T+1$ values. Then, $d$-order co-occurrences are computed, thus obtaining $(2T+1)^d$ entries, which can be significantly reduced by exploiting symmetries. Several $T$ and $q$ values, and several orders $d$ for the co-occurrences are considered in [59] to build the SRM set, consisting of 34.671 features in total. The SRMQ1 set, with reduced complexity, is obtained by fixing $q = 1$, and consists of 12.753 features.

The Subtractive Pixel Adjacency Matrix (SPAM) model [69] was the first residual feature set proposed. SPAM features are extracted as described above, by considering only the first order linear residual (difference array) in all the directions (horizontal, vertical, diagonal), then truncating the values at $T = 3$ (default) and finally computing the second-order co-occurrences ($d = 2$), for a final number of 686 features.

The SPAM set is the simplest set of rich features, rich enough to work well for a wide variety of tasks, and successfully used in the literature first in steganalysis and then in forensic applications [64, 66]. In this thesis we make extensive use of this feature set. Another advantage of the SPAM set is its reduced dimensionality, that allows standard training of SVMs without needing to resort to more complicated architectures and ensamble classification.

### 2.4.2 CNN-based image forensics

In the last decade, DL techniques and CNNs in particular have been successfully used for various image recognition and classification tasks [43]. In recent years, they have also been used extensively for applications of steganalysis [70–72], and many researchers have started exploring their use for multimedia forensic applications. In most cases, the performance of the new CNN-based techniques greatly exceed those of classical model-based and standard ML-based methods.

In most of the CNN models, the feature extraction process is carried out in a fully automatic way, and then the features are self-learned by the network from the input image. In some cases, the forensic designer can work on the architecture by properly designing the first layers in order to force some pre-processing steps that facilitate the learning of discriminative features.

One of the first works using CNNs for Image Forensics is the median filtering detector proposed in [73]. In [8], the authors proposed both a binary and a multi-class CNN, effective for the detection of several manipulation operations, that is, blurring, noise addition, resizing and median filtering (the work has recently been extended in [74]). CNN architectures for camera model identification have been proposed in [75] and [76]. The above mentioned architectures are rather shallow, consisting of only three or four convolutional layers. Moreover, a pre-processing filtering step (constrained first layer) is applied to the images to force the network to look for the artifacts in the residual domain high-pass image, thus facilitating its job (expectedly, the traces are better exposed in the residual image).

Recently, in [77], the authors proposed a novel method for aligned DJPEG detection and forgery localization that relies on a one-dimensional CNN, designed to automatically learn discriminant features from the histograms of

DCT coefficients. The approach outperforms those based on standard SVMs and hand-crafted features (e.g. the one in [48]), especially on small sized regions (up to 64 × 64). A more general approach for DJPEG detection based on CNNs, that works for both aligned and non-aligned DJPEG, that can then be exploited for forgery localization (due to the capability to work on small patches, i.e., 64 × 64 pixels), has been recently proposed in [78], where the capability of CNNs to capture DJPEG artifacts directly from the image pixels have also been explored. A method for the estimation of the DCT coefficients of the primary quantization matrix based on CNNs been proposed in [79]. The method can significantly outperforms state-of-the-art methods based on statistical analysis (e.g. in [32]), being capable to work on small window sizes and under very general operative conditions (under both aligned and non-aligned DJPEG, when the primary quantization is either stronger or weaker than the second one).

Recently, the trend in the Machine Learning community towards the adoption of very deep models has started been followed in multimedia forensics [79–83]. In many cases, the approach of considering completely self-learned features, without constraining the initial layers has proven to be beneficial, as long as enough training data is available.

### 2.4.3 Datasets

As we said, data-driven methods require that large scale datasets are used for training and testing. Below, we present two of the most common publicly available datasets used for ML applications in forensics.

**RAISE dataset** [**84**]**.** A dataset of camera-native uncompressed images in raw format. The entire dataset consists of 8156 images in total (RAISE-8K dataset). Three different devices (a Nikon D40, a Nikon D90, and a Nikon D7000) are employed, and the images are taken at very high resolution (3008×2000, 4288×2848, and 4928×3264 pixels) and saved in an uncompressed format as natively provided by the employed cameras. All the images have been collected over a period of 3 years, in over 80 places in Europe and are representative of a large number of categories (outdoor, landscape, nature, people, buildings, indoor and objects). The RAISE-2K dataset is a subset that contains a selection of 2000 (out of

8156) images. The majority of these images comes from Nikon D90 and have size 4288 × 2848.

**VISION dataset [85].** This dataset consists of 34,427 images in total, both in the native format and in their social version (Facebook, YouTube, and WhatsApp are considered). The native images (11,732) have been acquired by smartphones and tablets belonging to several brands (35 different mobile devices are considered). The images from VISION dataset are in JPEG format compressed with the defaults qualities of the acquisition devices. Several image resolutions are considered, depending on the mobile device, and ranging from 2560×1920 (lowest resolution device) to 5248×3936 (highest resolution device).

### 2.4.4 Discussion

The very good performance that can be achieved through the use of ML techniques comes with a number of drawbacks. ML, and in particular DL, suffers from a number of shortcomings hindering the application in the multimedia forensic scenarios: among them, the most relevant are the need for a large amount of (huge in DL) representative data and the general lack of security. Gathering large datasets which are representative of the variety of situations that can be encountered at test time is in fact a very difficult task for multimedia forensics applications. Moreover, the easiness with which it is possible to disable ML tools and in particular DL tools, as stressed by recent research in the field [14], makes it very hard to exploit these tools in a multimedia forensic scenario and more in general for security-oriented applications, where the possible presence of an adversary can not be ignored. This calls for the study and the development of a new class of ML-based forensic tools that can overcome the above problems, while still leveraging on the superior capabilities of modern ML technologies, which is the mission this thesis contributes to.

Chapter 3

# Introduction to Adversarial Image Forensics

*"Everything is a self-portrait. A diary. Your whole drug history's in a strand of your hair. Your fingernails. The forensic details. The lining of your stomach is a document. The calluses on your hand tell all your secrets. Your teeth give you away. Your accent. The wrinkles around your mouth and eyes. Everything you do shows your hand."*
Chuck Palahniuk

Together with the development of forensic tools for retrieving information and detecting possible tampering, in recent years, a large number of counter-forensics (CF) tools have also been developed to prevent a correct analysis. Such tools are often effective due to the fragility of forensic tools, which most of the times are not thought to work under adversarial conditions. However, in real-world applications, the presence of an adversary aiming at impeding the analysis cannot be neglected, therefore, developing forensics techniques capable to work in adversarial environment has become a necessity. This turns out to be a difficult task due to the weakness of the traces the forensic analysis relies on. The task of developing effective forensics techniques in adversarial setting is even more challenging when ML-based, and in particular DL-based, forensic tools are adopted for the analysis, due to the inherent vulnerability of these tools, and the reduced performance obtained when they are tested under different, or *mismatched*, conditions, with respect to those used for the training. This calls for the development of more secure ML-based forensic tools, namely, a new class of tools that can effectively counter CF attacks, at the same time keeping the benefits of modern machine learning techniques. The problem of the security of machine learning systems in adversarial environment is common to many other security-sensitive applications [86]; then, reasonably, the use of similar solutions should be investigated to secure image forensic techniques.

In this chapter, we first briefly introduce Counter Forensics and Anti-Counter Forensics, and review the state-of-the-art in the field. A terminology typical of Adversarial Machine Learning is adopted in our overview to formally define the general attack model and classify the CF methods developed so far. Then, the solutions proposed so far to combat CF attacks are discussed adopting the perspective of the analyst, by adopting the classification introduced in [87]. This classification is an interesting and general one, that also guides the classification of the defence (anti-CF) approaches (to secure the forensic analysis) proposed in this thesis, and presented in the next chapters.

## 3.1 Counter-Forensics and Anti Counter-Forensics

Counter-forensics, also referred to as anti-forensics, refers to all the solutions devised to bypass the forensic analysis.

The origin of Counter-Forensics (CF) traces back to [88] where the concept of fighting against Image Forensics was first introduced. Early proposed CF techniques were rather simple, consisting in the application of some basic processing operators, see for instance [89–91]. In [91], the gaps in the DCT histograms are reduced by spreading the coefficients by means of additive noise (dithering), and the blocking artifacts are hidden by means of a smoothing operation, to hide JPEG compression. Similarly, the method developed in [90] removes picks and gaps in the pixel histograms through dithering, to conceal a contrast enhancement operation. In [89], image high frequencies are perturbed with noise while being resampled, to conceal traces of resampling. These techniques can be easily circumvented by dedicated methods, as those described in [92–94].

When the attacker has some information about the forensic algorithm, more effective CF techniques can be devised. In most of the cases, CF techniques are tailored to attack a specific algorithm. Such techniques are referred to as *targeted attacks*, without paying attention to the possible countermeasures adopted by the analyst, e.g. by neglecting the fact that the CF attacked images can be later subject themselves to forensic analysis. CF methods, in fact, are not perfect and leave traces on their own, that can be exploited by an informed analyst. The techniques developed by the analyst to defend against CF attacks are referred to as *Anti-Counter-Forensic* techniques (see Section 3.3.).

## 3.2 An Overview on Counter-Forensic Attacks

We start by introducing some useful terminology, as proposed in [86] and later in [95], to categorize the attacks in machine-learning, and the necessary formalism.

According to the taxonomy in [86,95] , the attacks can be classified based on several properties: the type of influence the attack has on the system (*causative* or *explorative*); the specificity of the attack (*targeted* or *indiscriminate*); the kind of security violation the attacker aims at (*integrity* or *availability*). In particular:

- **Influence.**
  - *Causative*: attacks carried out at training time.
  - *Explorative*: attacks carried out at test time.
- **Specificity.**
  - *Targeted*: when the attack focuses on the deception of a specific algorithm (classifier).
  - *Indiscriminate*: when the attack has a more flexible goal, or when the attack is targeted to a class of algorithms (rather than a specific algorithm).
- **Security violation.**
  - *Integrity*: when the attack aims at letting malevolent samples be classified as normal (false negative error).
  - *Availability*: when the attack aims at causing a classification error of any type, i.e., both a false negative and a false positive error, thus causing a denial of service (DoS).

The above taxonomy can be used to define the attack scenarios and to classify the attacks, as done in [96, 97], for attacks to general machine learning and pattern recognition systems. Below, we provide some useful notation and formalism.

Without loss of generality, we focus on the case of binary classification. Let $H_0$ be the hypothesis that the image under analysis is pristine, and $H_1$ the hypothesis that the image is manipulated, or, more in general, it contains the trace the forensic analyst is looking for. When a manipulated sample is erroneously classified as pristine (i.e, the decision is in favor of $H_0$ when $H_1$ holds), a *false negative* of *missed detection* error occurs. When instead a pristine sample is erroneously classified as manipulated (i.e, the decision is in favor of $H_1$ when $H_0$ holds), we have a false positive of false alarm error.

In the following we stick to the following formalism. We denote with $\phi$ the forensic algorithm used by the analyst, or, simply, the detector. $\phi$ depends on: i.) the type of algorithm, its structure and parameters $l_i$ (as well as the learning algorithm, for data-driven methods), all together denoted by $\mathcal{L} = \{l_1, l_2, ..\}$; ii.) the feature space $\mathcal{X}$; iii.) the training data $\mathcal{D}$ (for data-driven approaches only). Therefore, $\phi(\mathcal{L}, \mathcal{X})$ in the model-based case, and $\phi = \phi(\mathcal{L}, \mathcal{X}; \mathcal{D})$ for ML-based algorithms. In the sequel, we refer to $\phi$ as $\phi(\mathcal{L}, \mathcal{X})$, the dependence on the training data being explicitly stated only when needed. When useful, the CF method adopted by the attacker is indicated by $A$.

### 3.2.1 Counter-forensic attack model

In this section, we formalize the general adversarial model for CF attacks. By following [96, 97], an adversarial model is defined by specifying the assumptions about adversary's goal, knowledge of the system, and capability to modify the data and corrupt the system.

**Attacker's goal**: the attacker's goal specifies the kind of security violation, hence the kind of error the attacker aims at. CF attacks are usually *integrity violation* attacks or evasion attacks: the attacker wants to modify the manipulated images ($H_1$ class) so that they are misclassified by the detector, hence deemed as pristine ones ($H_0$). In doing so, he normally wants to inject a small distortion into the image (ideally, the minimum distortion allowing him to cross the decision boundary of the classifier), so to keep the visual distortion low. The adversary's goal determines the loss function that the adversary seeks to maximize.

**Attacker's knowledge**: by following the terminology in [98], the attack can be Perfect Knowledge (PK) or Limited Knowledge (LK). In the PK scenario, the attacker has complete information about the forensic algorithm. This is the most favorable case for the attacker. In the LK scenario, instead, the attacker knows only some details about the forensic algorithm: e.g., he does not know the exact algorithm or some of the parameters of the algorithm $l_i \in \mathcal{L}$, or, for ML-based methods, he does not know (or knows only partially) the training data $\mathcal{D}$. Although indirectly, the amount of knowledge available to the attacker about the forensic algorithm determines the specificity of attack. In most of the early attacks in CF, it is (implicitly) assumed that the forensic algorithm is fully known to the attacker, hence a *targeted attack* is launched to the classifier. An obvious drawback of CF techniques designed to attack a specific algorithm is that they neglect the possible reaction of the analyst and the countermeasures that he could adopt to prevent the attack. Recently, there was also a trend towards the development of more general approaches, referred to as *universal*, that are designed in such a way to be effective against an entire class of forensic classifiers [99, 100].

**Attacker's capability**: it is most suited for machine-learning methods, and refers to the amount of control that the adversary has on training and testing data. Therefore, the attacker's capability is related to the influence of the attack. In the *exploratory* attack scenario, the adversary's capability is limited to modifications of test data, and altering the training data is not allowed. Viceversa, in the a *causative* attack scenario, the attack can interfere with the training process; these attacks are usually referred to as poisoning attacks. Most of the CF attacks proposed so far belong to the category of *exploratory* attacks.

The above definition of the attack scenario, that allows to clearly formalize the threat model for the problem under investigation, is of major importance not only to classify the attacks, but also to help the analyst, or more in general, the system designer, to device proper techniques that can work in an adversarial environment (see Section 3.3).

As we said, the large majority of the CF attacks proposed so far are *exploratory evasion* attacks [98]. In the following, we provide some representative examples for the various attack models. The review of attacks against deep learning forensic methods, which are very recent and still pretty limited at the time of writing, is discussed later in a dedicated Section (3.2.4).

### 3.2.2 Attacks with perfect knowledge

In the perfect knowledge scenario, the attacker can build the attack by relying on the knowledge of the forensic algorithm $\phi$. Then, a targeted attack can be launched to the system [101]. In this case, it is possible for the attacker to induce an integrity violation, i.e., inducing a false negative decision error, by introducing a limited, ideally minimum, distortion, thus preserving as much as possible the quality of the attacker image. Generally speaking, the attacker needs to solve an optimisation problem looking for the image which is in *some sense* closest to the image under attack, among those for which the output of the forensic algorithm is the wrong one. Although solving the optimization problem is often not easy, the knowledge of the algorithm $\phi$ permits to devise very powerful CF methods. This is the case of the CF method in [102] for countering the model-based detectors of double (multiple) JPEG compression based on the analysis of the First Significant Digits (FSD), or the general approach in [103], where the optimal attack against Benford's law-based detectors is derived. The approaches in [104] for hiding the traces of median filtering in digital images and the one in [105] to counter SIFT-based copy move detection are other examples of closed form CF methods.

When the detector is more complicated, as it is often the case with machine learning-based approaches, the optimum attack can be implemented by relying on *gradient-descent* solutions, see for instance [7], where a gradient-based attack in the pixel domain is proposed and applied to counter SVM manipulation detectors based on residual features, or [106], where a suboptimum approach based on gradient analysis is proposed to counter forgery detection.

A problem with many PK approaches in the CF literature is that the algorithm is directly applied in the feature domain, and then the distortion is controlled (or minimized) in this domain. However, in this way, it is hard to control the distortion finally introduced in the pixel domain, all the more that the relationship between the pixel and feature domain is often non-invertible, thus also raising the problem of mapping back the attack (e.g., in [102]). When the relation between the feature domain and the pixel domain is invertible, as it is the case with several transformed domains (e.g. the DCT domain), the image distortion can be controlled by operating in the feature domain (e.g. by resorting to *optimal transport* [107, 108], as it is the case in [99, 100]). In general, when the relationship is more complicated, a suboptimum strategy is to implement the attack in two steps: first the attack which minimises the distortion in the feature domain is determined, then a new minimisation is carried out in the pixel domain in order to get close to the desired attack, as done in [106]. A gradient-based attack directly applied in the pixel domain, which then does not require invertibility of pixel and feature domain, is the one proposed in [7].

### 3.2.3 Attacks with limited knowledge

We consider the taxonomy introduced in [87], and already used therein to classify the CF attacks within this category.

**Universal attacks**. The attacker *only* knows the feature space or class of features $\mathcal{X}$, adopted by the forensic detector. Since he is not aware of the exact statistics used by the analyst, he performs an attack which is effective against *any* detector $\phi'$ inside the class $\Phi = \{\phi(\mathcal{L}', \mathcal{X}), \forall \mathcal{L}'\}$.

Some examples of attacks referring to the first category are the *universal* CF methods in [99] and [109], developed versus the class of detectors based on first-order statistics of the image. The method in [99] is applied to deceive contrast enhancement detectors. Similarly, in [100], a universal method against the class of detectors based on first order statistics in the DCT domain is proposed, and used to counter the detection of double (multiple) JPEG compression. Noticeably, the detectors based on the FSDs of DCT coefficients (e.g. [53]) are particular instances of the first order DCT domain class.

**Attacks based on a surrogate detector**. In this case, the attacker has a partial knowledge of the algorithm $\phi$; for example, he might know the feature space $\mathcal{X}$ but not all the parameters of the algorithm $\mathcal{L}$ and/or, in the case of ML-based methods, the training set $\mathcal{D}$. In this case, the attacker generates a *surrogate* detector $\hat{\phi}$ (sometimes called *substitute* detector), by exploiting the information available to him and making an educated guess about the parameters he does not know. Then, he builds the CF attack by performing a targeted attack against $\hat{\phi}$, hoping that the attack will also work against the real detector (*attack transferability*) [13]. Formally, if we let for instance $l_1$, $l_2$ be the unknown parameters, then $\hat{\mathcal{L}} = \{\hat{l_1}, \hat{l}_2, l_3, l_4...\}$ where $\hat{l_1}$ and $\hat{l}_2$ are the attacker's guesses of $l_1$ and $l_2$ and $\hat{\phi} = \phi(\hat{\mathcal{L}}, \mathcal{X}; \mathcal{D})$. The effectiveness of the method is then assessed against the original $\phi$ detector.

An example of attack based on a surrogate detector is the famous fingerprint-copy attack for PRNU-based camera identification [110]: the real camera fingerprint $K$ (namely, a parameter of the classifier, $K \in \mathcal{L}$) is unknown to the attacker, who then bases the attack on an estimation of $K$, $\hat{K}$, made by relying on a set of available images drawn from the camera. Many attacks to ML detectors fall into this category: in fact, even if it is often safely assumed that the attacker knows the kind of classifier used (e.g., an SVM, or a neural network), and its architecture, he rarely has access to the same dataset $\mathcal{D}$ used by the forensic analyst to train the classifier. Reasonably, in this case, the attacker builds another dataset $\hat{\mathcal{D}}$, sampled from the same distribution, and uses it inside of the real one, thus attacking an home-made replica of the detector $\phi(\mathcal{L}, \mathcal{X}; \hat{\mathcal{D}})$ (see for instance [7], and [98, 111] from the general ML literature). Another LK attack for the case where the attacker knows only the feature space $\mathcal{X}$ and guesses both $\mathcal{L}$ and $\mathcal{D}$ is provided in [106]. It is worth stressing that such attacks work well under the assumption of *attack transferability*, that is, assuming that the attack maintains the effectiveness (at least in part) even against algorithms, different from the one targeted by the attack. Noticeably, standard ML tools are known to be sensitive to the problem of *database mismatch* [112], then, relying on home-made replica of ML classifiers is not always effective to build an attack which works against the targeted classifiers.

**Laundering-type attacks**. The attacker has only a very general and limited knowledge concerning the algorithm; then, he attempts to erase the traces of CF by applying some basic processing operations (e.g., filtering, resampling, and etc...). In this case, the attacker does not target any specific detector or class of detectors.

Early CF techniques developed against detection of resampling [89], single and double JPEG compression [91, 113], contrast adjustment [90], median filtering [114], and splicing detection [115], can be categorized as *laundering-type* attacks. We already mentioned many of them in Section 3.1. It is worth noticing that this way of categorizing the above attacks is quite recent. In the literature, such attacks are often referred to as targeted attacks, then they could be included in the PK category. The reason why these approaches are not included among the PK attacks by [87] is the following: in that methods, the knowledge of the algorithm is 'only' marginally exploited (even if it is available), and they boils down to the simple application of a basic processing, which turns out to be effective to the purpose, without kind of optimization. In most cases, the specific algorithm $\phi$ is only used to prove the attack effectiveness, and not to guide the attack. Though very simplistic, the application of a post-processing operation has recently been shown to be very effective also against general SVM-based manipulation detectors trained on rich features models [66].

A noticeable strength of such CF attacks with respect to most PK attacks is that they are much easier to implement. Furthermore, by applying a basic processing, the attacker can easily control the distortion introduced into the image.

### 3.2.4 Attacks to deep learning-based image forensics

Given the recent trend in Image Forensics towards the use of deep learning architectures, CF attacks to deep learning models have started been developed as well in the last couple of years.

A key advantage of CNNs is their ability to learn forensic features directly from the input data, that is, directly from the image. On the other hand, an intelligent attacker can use this property to his advantage, thus enabling new kinds of CF attacks. The vulnerability of deep learning has recently been studied by many works, also in forensics [116]. The main underlying motivation for the vulnerability of CNNs for image forensics, and more in general image recognition applications, is the following: since the space of possible inputs to a CNN is substantially larger than the set of images used to train it, an attacker can easily create modified images that fall into an 'unseen' regions of the image space and thus force a misclassification error.

The so called *adversarial examples*, originally introduced in [117], exploit the above property to build very small (often quasi-imperceptible) perturbations of the input image, which are sufficient to cause an incorrect decision. In this way, an attacked image is produced which is visually indistinguishable from the original one, but is misclassified by the CNN. These perturbations are typically learned by computing the gradient of the loss function with respect to the input image as done in the Fast Gradient Sign Method [14], and DeepFool attacks [118], or by using iterative methods such as the Jacobian-Based Saliency Map Attack [119], or the box constrained L-BFGS [117], that approximate the gradient descent solution. The above mentioned adversarial examples to DL models are attacks carried out at test time, and then belong to the category of *exploratory* attacks; moreover, they assume that the attacker has access to the CNN model (PK scenario). By following the specific taxonomy for adversarial examples introduced in [120], the *PK scenario* is often referred to as *white-box* scenario in DL literature. Similarly, the LK scenario is referred to as *gray-box* scenario [121]. Black-box attacks are also considered in the DL literature where the internal details are not available to the attacker, which is a more realistic and challenging scenario; therefore, he applied several times queries to gain the internal details.

CF adversarial examples have been proposed against CNNs in multimedia forensics. The first targeted attack based on adversarial examples was proposed in [122] to fool a CNN-based camera model identification system. In [123], adversarial examples against CNN-based methods for several common manipulations, and among them blurring, resizing and median filtering, have been proposed. Adversarial perturbations have been shown to offer poor robustness to image processing operations: in particular, the operation of rounding to integers is sometimes already sufficient to wash out the perturbation and make an adversarial example ineffective [116]. A gradient-inspired pixel domain attack to produce adversarial examples in the integer domain has been proposed in [124], and has been applied against CNN-based detection of resizing, median filtering and contrast adjustment.

Being targeted to a specific CNN detector, CF adversarial examples are PK attacks. Thus the counterfeiter hopes that the attacked image produced in this way can also work against other CNN detectors, as it is the case with common DL applications of computer vision (pattern and image recognition), where the attack transferability assumption holds under a wide variety of scenarios [13, 14].

Together with adversarial examples, CF researchers have started considering the use of Generative Adversarial Networks (GANs) [125]. GANs are a learning framework developed to create generative models capable of statistically mimicking the distribution of training data. This is done by iteratively training a discriminator to differentiate between real and generated samples of data and training the generator to produce samples capable of fooling the discriminator. GANs have been used by the computer vision community to produce visually realistic images or super-resolution versions of images. A GAN capable of removing forensic traces left by median filtering has already been developed in [126]. Inspired to the work in [126], a GAN to perform anti-forensics of JPEG compression has been proposed in [127]. CF attacks obtained via GANs are PK attacks that target a specific forensic detector, namely the discriminator.

## 3.3 Anti-Counter Forensics Methods

As a reaction to CF, several anti-CF methods have been developed to restore the validity of the forensic analysis. Most of these approaches are tailored against specific CF methods. By following [87], we adopt the viewpoint of the analyst to classify the solutions proposed so far to counter CF attacks. In particular, we distinguish between *adversary-aware systems*, *intrinsically more secure detectors.* and *solutions based on game-theoretic analysis.*

### 3.3.1 Adversary-aware detectors

This is the most common approach used so far. The analyst, who is assumed to be aware of the CF method the system is subject to, develops a new algorithm capable of revealing the attack, by looking for the specific traces left by the CF tool. In most cases, this goal is achieved by resorting to new, tailored, features. Then, a new algorithm $\phi_A$ is explicitly designed to reveal if the document underwent the CF attack, and used in conjunction with the original, unaware, algorithm $\phi$. This is the case, for instance, of the approaches adopted in [94, 128–130], to address respectively adversarial detection of JPEG compression [94, 128], median filtering [129] and splicing [130]. Among other examples, we mention the algorithm proposed in [131] for defeating the fingerprint-copy attack to PRNU-based camera identification and the one in [132], to combat the SIFT-keypoint removal and injection attack against copy-move detectors. In other cases, the new algorithm is obtained by using the same features of the original algorithm $\phi$, and designing an adversary-aware version of the algorithm $\phi_A$, which is then used in place of $\phi$. Obviously, this approach is particularly suited for ML-based approaches, where the original algorithm can be re-trained by including also examples of attacked images in the training set. In this way, the analyst obtains a refined detector $\phi_A = \phi(\mathcal{L}, \mathcal{X}; \mathcal{D} \cup \mathcal{D}_A)$, where $\mathcal{D}_A$ is the set of attacked images used for training. In general, this approach is viable when the feature space is discriminative enough, i.e., it is capable to distinguish original, manipulated, as well as attacked images. This is the case of general residual-based features (e.g. the Subtractive Pixel Adjacency Matrix (SPAM), [69]), or, even better, rich features models [59]. Examples of this approach can be found in [133] for adversarial double compression detection, and in [134], for a variety of manipulation detection problems under JPEG laundering attack.

Recently, exploiting the superior capabilities of deep learning architectures to learn feature representations, adversary-aware training has widely been used to improve robustness of DL models to adversarial examples [14].

The performance of the adversary-aware systems are assessed by (implicitly) assuming that the attacker keeps attacking the original, unaware, algorithm $\phi$. Therefore (by using the terminology introduced in the previous section for the attacks), by adopting the above approach, what the analyst does is trying to exit the PK scenario.

### 3.3.2 Generally more secure detectors

As pointed out before, most of the anti-CF techniques are developed without taking into account the possibility that the attacker foresees the move of the analyst and react properly. Obviously, when the attacker anticipates that the traces left by the CF tools may themselves be subjected to a forensics analysis, then he may refine the counter-measures devising more powerful CF techniques that leave less evidence into the forged documents. This obviously leads to an unavoidable loop, wherein CF and anti-CF techniques are iteratively developed (the so called 'cat& mouse loop'), whose outcome can hardly be foreseen [135, 136]. A possible approach to avoid this problem is to design the forensic techniques in such a way that they are intrinsically more resistant to CF attempts and then, more difficult to attack even in the PK case. In this case, then, differently from the previous case, the analyst does not specialize the algorithm to resist to a particular CF tool.

Improved intrinsic security can be achieved in several ways. A possibility is to use higher order statistics; formally, being $\mathcal{X}$ the set of features of the original algorithm, the algorithm is refined by considering larger feature spaces $\mathcal{X}'$, where then $\mathcal{X}' \supset \mathcal{X}$. This is done for instance in [137, 138], where second order statistics based on co-occurrence matrices are considered for the detection of contrast enhancement in presence of attacks. A similar approach is adopted in [50, 139, 140] for countering, respectively, JPEG, double JPEG and local tampering anti-forensics. In all these cases, resorting to second-order statistics allows the analyst to expose CF attacks and re-establish a correct analysis. Another possible approach to design an intrinsically more secure system consists in fusing the outputs of several forensic algorithms looking for different traces [141].

More in general, approaches belonging to this category look for solutions that can work under the (or a kind of) worst-case attack, for a given class of attacks. For ML approaches, this can be achieved by means of aware training, considering examples of such worst-case attack $A^*$. An example of such a method can be found in [133], for the detection of double JPEG compression, where the original algorithm is refined by training on $\mathcal{D} \cup \mathcal{D}_{A^*}$ where $A^*$ is the optimum attack to first order statistics in the DCT domain. Another possibility is to resort to intrinsically *more secure* features, as done in general literature about ML security, by optimizing in some way the feature set, for instance by looking for the best feature set (starting from a large feature space) against a PK attack [111], or searching for intrinsically *more secure* architectures [97]. Randomizing the feature selection according to a secret key, thus preventing the attacker from gaining full knowledge of the system, is another way to design a more secure algorithm; such a strategy has been proven to be effective against PK attacks to SVM-based detectors [16]. Other randomization selection strategies have been proposed in many security-related areas (e.g., steganalysis, spam filtering and intrusion detection) but they have not been applied in forensics.

### 3.3.3 Game-theoretic solutions

A possible approach to stop the never ending loop leading to the continuous development of CF and anti-CF techniques, is to resort to *Game Theory* [142, 143]. Game theory provides a way to model the interplay between two rational players, namely the attacker and the forensic analyst, and tools that can be used to study the interplay and then identify the optimum attack and detection strategy for the attacker and the analyst [135, 136]. Such optimum strategies are obtained by searching for the equilibrium point of the game, namely a point that represents a satisfactory solution for both the contenders, from which then neither the analyst nor the adversary has interest to deviate. To elaborate, forensic games are typically formulated as two player games [142, 143], where the analyst's utility is defined as the probability of detecting a forgery and the attacker's utility is defined as the probability that the forgery is not detected.

By considering the strategy at the equilibrium of the game, the forensic analyst can design an intrinsically secure forensic algorithm, at least in the adversarial scenario considered for the analysis.

Game theoretical approaches for forensic tasks have been developed in [144–146] for model-based analysis, and in [147, 148] for system based a training data. Though rather theoretical in nature, these works provide a natural framework to cast multimedia forensics in and can provide very useful insight about the achievable security of a wide class of multimedia forensic tasks [149].

A limitation with the game-theoretic approach is that it is often not easy to define a 'solvable' game for the problem under analysis (that is, a game that can be solved using game theoretical tools and analysis). This is especially true when the interaction between the forensic analyst and attacker is a non trivial one, as it is often the case in multimedia forensic.

Chapter 4

# Higher-Order, Adversary-Aware, Double JPEG-Detection via Selected Training on Attacked Samples

*"If you have built castles in the air, your work need not be lost; that is where they should be. Now put the foundations under them."*
Henry David Thoreau, Walden

This chapter is devoted to the design of an adversary-aware detector capable to reveal if an image has undergone a double JPEG compression, possibly in the presence of other processing or generic counter-forensic attacks. Such a detector is obtained by training an SVM classifier in an adversary-aware manner, including a limited number of properly selected examples of attacked images. The chapter is organised as follows: the reasons behind this research are given in Section 4.1. Then, in Section 4.2 we describe the general idea behind the proposed detector and our choice of the set of features for the adversary-aware classification (a quick introduction on related rich feature models is also given in Section 2.4.1.2). The experimental campaign and the results of the experiments are discussed in Sections 4.3 and 4.4.

## 4.1 Motivation and Contribution

A practical problem with many defence strategies for double JPEG detection, as well as for other forensic detection tasks, is that they work against a specific attack, or class of attacks. When other kinds of attacks are considered, however, these techniques are expected to fail, all the more that the attacker may decide to combine his attacks with laundering-type processing, that is, geometric transformations or other processing capable of impeding a correct detection. All the more that, in real applications, the attack is not known in advance, thus impeding to build an ad-hoc detector. This problem is particularly relevant with data driven detectors based on machine learning due to the difficulty of training the detector on all possible attacks. In order to alleviate the above problem, we built an SVM classifier based on a large number of features computed both in the pixel and the frequency domain and add to the training set some images which underwent a limited set of attacks. Using such a large number of heterogeneous features ensures that the classifier has the necessary degrees of freedom to distinguish images processed in several different ways. The use of features computed in the pixel domain is motivated by the need to cope with geometric attacks that de-synchronize the $8 \times 8$ grid at the basis of JPEG compression. With regard to the attacked images used to train the adversary-aware version of the classifier, we include only images processed with the attacks that, when used against a non-aware version of the classifier, result in the worst performance. The rationale behind such a choice is that a detector trained to recognise images subject to this kind of *Most Powerful Attacks (MPAs)* should also be able to detect double compressed images subject to milder processing.

It is worth stressing that, although in our research we focused on double JPEG detection, the arguments about the MPA-aware classification are general and can be applied to other decision tests under adversarial conditions. The interest of forensic researchers in the detection of double (and also multiple) JPEG compression is motivated mainly by the fact that JPEG is the most widely adopted compression standard, and when JPEG images are manipulated by an attacker, double compression often occurs (since the image is re-saved in JPEG format after the manipulation). Then, a double compression is an indirect of indication that a manipulation has taken place.

## 4.2 MPA-Aware SVM Detector

The idea behind our approach is illustrated in Figure 4.1: training on benign samples leaves wide room for attacks. Adding attacked samples to the training set permits to refine the decision region and make new attacks more difficult. Since it is not viable to consider all possible kinds of attacks, we train the

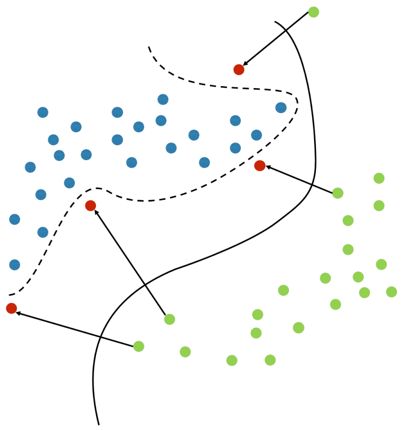


Figure 4.1: Rationale behind the design of the adversary-aware classifier. The introduction of a limited number of attacked samples (red dots) permits to narrow the region around legitimate samples (blue) thus making more difficult to camouflage green samples as blue ones.

classifier by including only those attacks that degrade most the performance of an unaware version of the classifier. Hereafter we refer to such attacks as MPAs, and the detector trained to recognise them MPA-aware detector. When the analysis is limited to first order statistics and the attack must satisfy a per-pixel distortion constraint, the optimum attack is known and the MPA corresponds to this attack (see [99] and [100] for attacks in the spatial and frequency domain respectively). The optimum attack in the DCT domain has been used in [133] to build an adversary-aware SVM for double JPEG detection, which was shown to be able to resist to double JPEG counter-forensic attacks belonging to the same class, namely, first order attacks (e.g., the attack to the FSD coefficients [102, 150]).

Exploiting the MPA-aware approach, we overcome the first order statistic limitation inherent in the analysis proposed in [133], and build an adversary-aware detector which is able to work under a wider variety of attacks. As a second goal, we also aim at improving the resilience against attacks, like geometrical attacks, for which the visual distortion introduced cannot be measured (and hence constrained) on a per-pixel basis.

Figure 4.2 schematises the detection task addressed in this research: we

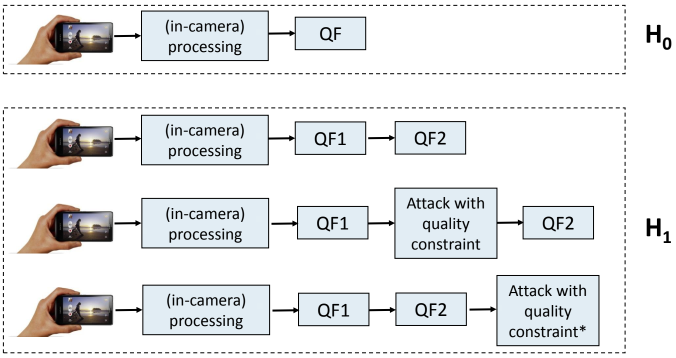


Figure 4.2: Adversarial DJPEG detection task.

let $H_0$ correspond to the case of single compressed images (in the absence of manipulation), and $H_1$ to the case in which the image is either compressed twice or compressed, attacked and then compressed again. An attack placed in the middle between the two compression stages may correspond to the application of a processing operation or to a CF attack to single compressed images, i.e. an attack aimed at erasing single compression traces so to make the image look like an uncompressed one. When the attack occurs after the second compression (last row in Figure 4.2), we implicitly assume that it ends up with a JPEG image. This is the case of a CF attack aiming at making a double compressed image look like a single compressed one. We observe that a three class classification could also be considered to distinguish between single compressed, double compressed, and double compressed and attacked images. However, we opted for a two-class approach since our purpose is to use the presence of double JPEG traces as an indication that the image has been processed in any way after its acquisition. The rationale behind our approach is that most images are stored in JPEG format, and hence any processing is always accompanied by a double JPEG compression.

In order to build a classifier capable to capture different types of dependencies among neighboring pixels, we need to resort to a large number of features. A possibility would be to adopt the rich feature models for both spatial and frequency domain described, respectively, in [59] and [151]. However, the huge dimensionality of these models asks for an extremely large training set making the use of standard machine learning techniques, like the SVM, no longer viable. By following a common trend in the literature, we select only some higher-order features (both in spatial and frequency domain) to build a model which is rich enough to capture the artifacts introduced by DJPEG compression under various attacking conditions, for which SVM classification is still feasible. We selected the SPAM features, described in Section 2.4.1.1 and 2.4.1.2, for the pixel (spatial) domain, and the CC-PEV features [152] for the DCT domain. In the CC-PEV model, we considered the global histogram and individual histograms for 5 DCT modes at low frequencies, total variation and blockiness (capturing the inter-block dependence) and transition probability matrix from difference arrays (capturing the inter-block dependencies). The final feature space dimensionality is 960, where 686 is the feature dimensionality for SPAM and 274 is the one for CC-PEV (with reference to [152], calibration is not considered for the features in the CC-PEV model).

## 4.3 Experimental Methodology

To create the datasets for our experiments, we started from gray-scale images in uncompressed format. Part of the images were used for training ($\mathcal{S}_{tr}$) and part for testing ($\mathcal{S}_t$). We built the images to be used for training and testing according to the following procedure (we refer to Figure 4.2): for the first class ($H_0$), the images were single compressed with quality factor $QF$[1]; for the second class ($H_1$), the double compressed images were built by first compressing the images with various $QF1$s and then with $QF2$. The attacked images were obtained by first compressing the same images with the $QF1$s, then attacking them with various processing and/or counter-forensics, and finally re-compressing them with $QF2$. For a meaningful analysis, we let $QF = QF2$. In summary, for each image, we built a single compressed version with $QF2$, many double compressed versions with second quality factor $QF2$

[1] The quality factor ($QF$) is used by many algorithms to set the quality of the JPEG image. A high quality factor corresponds to a high quality image (weak compression), a low quality factor to a low quality image (strong compression). More specifically, the quality factor determines the quantization matrix, i.e., the matrix of the quantization steps at the various DCT frequencies, used for the JPEG compression (see [153] for the details of the JPEG compression standard).

(one for each $QF1$), and the same number of attacked versions for each attack.

To compress and attack the images we used the Matlab image processing toolbox. Specifically, we considered geometric transformations (resizing, zooming, rotation, cropping, mirroring and seam-carving), filtering operations (median filtering, blurring, denoising), histogram enhancement and editing (copy-move). For resizing and rotation, we considered the bilinear (BIL), bicubic (BIC) and nearest-neighbor (NN) interpolation methods. Specifically, we considered a resizing scaling factor of 0.9, a zooming factor of 1.2, and a rotation angle of 5 degrees. Cropping was carried out by considering a $440 \times 440$ area both aligned and non-aligned with the $8 \times 8$ JPEG grid of the image. As to seam carving, the number of vertical seams to be removed was chosen in such a way that the final image has approximately the same size of a resized image with resizing factor equal to 0.9. With regard to filtering, In order to limit the visual degradation of the attacked images, we considered a $3 \times 3$ window for the median filter, and a $3 \times 3$ Gaussian smoothing kernel with variance $\sigma^2 = 1$ in the blurring operation. For denoising, we considered the wavelet-based filter proposed in [154] with $\sigma^2 = 10$. Histogram enhancement was performed by using contrast-limited adaptive histogram equalization (CLAHE) [54, 65, 155, 156]. Finally, in the copy-move operation, a random part of the image of size $256 \times 256$ was copied and pasted into a different part.

Regarding CF, we considered the anti-forensic JPEG algorithm described in [6], which removes the blocking artifacts of JPEG compression by applying a median filter followed by the addition of a Gaussian noise (*dithering*). Figure 4.3 depicts the scheme of the CF attack in [6].

Such a scheme is known to be quite an effective CF attack; however, its impact on the attacked image is perceptually significant (especially when the attack is applied in the DCT domain) [157]. To limit visual degradation, we considered a 3x3 median filter and a small variance $\sigma^2$ for the noise which is related to the variance of the image and ranges from 1 to 2. It is worth pointing out that, the CF scheme in [6] aims at erasing the traces of single compression, and then corresponds to a case in which the attack is placed between the two compression steps (last row in Figure 4.2). Finally, we also consider the universal counter-forensic schemes of proposed in [133], that is, the MPA against first order based detector.

To build the classifier, we used the (960-dimensional) features extracted from the images to feed an SVM with Gaussian kernel . The kernel parameters are chosen by 5-fold cross validation. In the unaware case, we trained the SVM with single and double compressed images and then tested it on single, double and attacked images. In the adversary-aware case, we trained the SVM also with examples of attacked images. To choose the attacks for aware training, we considered the attacks leading to the worse classification accuracy in the unaware case. In almost all our experiments we set $QF2 = 85$ and considered several values of $QF1 < QF2$. Some experiments for the case $QF2 = 95$ have also been performed and reported to show that similar results hold for a different choice of $QF2$, with some obvious differences in the numerical values expressing the performance of the detector.

## 4.4 Results

In our experiments, camera-native (uncompressed) images were taken from the RAISE dataset. We also used uncompressed images from the Dresden Image Database [158] and the VISION database for additional testing. Specifically, the 2000 images in RAISE-2K were split as follows: 1400 images were selected to build the training set (plus other 300 images used for setting the kernel parameters, i.e., internal cross validation) and 300 images for the test set. These images have a large size (4288 × 2848), then to fasten the feature computation, we sub-sampled them down to a size of 1072 × 770. Larger images of size 2144 × 1424 were also considered for testing.

### 4.4.1 Choice of the attacks for MPA-adversary aware training

We trained the unaware SVM classifier with the images in $\mathcal{S}_{tr}$ single compressed with $QF = 85$ and double compressed with $(QF1, QF2) = \{(50,85), (65,85), (70,85), (75,85), (80,85)\}$. Accordingly, the training set contained 1200 single compressed images for $H_0$ and 7000 (5×1200) images for $H_1$. Table 4.1 shows the performance of the unaware SVM. The Area Under the Curve (AUC) of the ROC curve for the classification single vs double and single vs attacked images (for all the considered attacks) is given. The results

Table 4.1: AUC values of the unaware SVM classifier.

| **Attack** | DJPEG | 1st order MPA | Stamm Attack | wavelet denoise | median filtering | copyMove 256x256 |
|---|---|---|---|---|---|---|
| **AUC** | 1 | 0.98 | 0.43 | 0.8 | 0.62 | 0.99 |
| **Attack** | CLAHE | resize BIC0.9 | resize BIL0.9 | resize NN0.9 | rotation BIC5 | rotation NN5 |
| **AUC** | 0.98 | 0.49 | 0.53 | 0.58 | 0.56 | 0.58 |
| **Attack** | zoom 1.2 | crop align | crop no align | mirror | blur | seam-carving |
| **AUC** | 0.64 | 0.72 | 0.70 | 0.58 | 0.86 | 0.97 |

for rotation BIL are not reported being always very similar to those of the BIC case. The unaware SVM is able to correctly classify single and double compressed images with great accuracy (AUC = 100%). Not surprisingly, the unaware SVM is also able to counter the double JPEG anti-forensic technique in [133]. Indeed, since the scheme is limited to first order statistics of the DCT coefficients, it leaves traces on higher order statistics. However, the unaware SVM fails to classify the attacked samples for almost all the manipulations. The most harmful attacks are the geometrical attacks and the counter-forensic attack in [6], hereafter denoted as StammAttack, in which case the unaware detector completely fails.

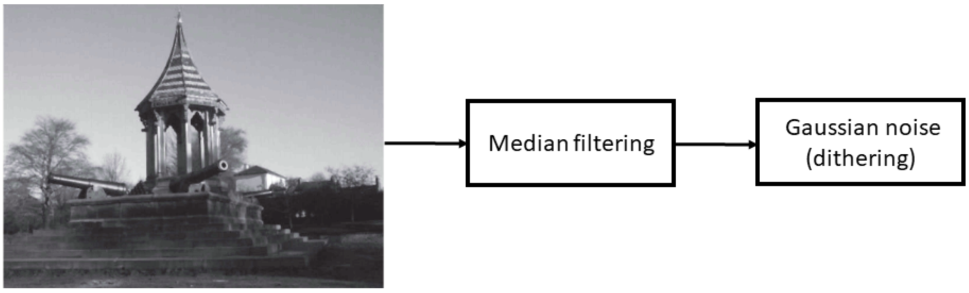


Figure 4.3: Scheme of the CF attack in [6].

### 4.4.2 Generalization capabilities of the MPA-aware detector

Based on the above results, we built the MPA example images by considering the resizing attack (with bicubic interpolation) and the anti-forensic attack by Stamm et al. [6], and re-trained the SVM by adding examples of images attacked in this way. In fact, these kinds of manipulations alter the image in a completely different way and training the classifier to recognise one of the two processing does not help with respect to the other. Accordingly, the images of the training set $\mathcal{S}_{tr}$ were compressed with $QF1 = \{50, 65, 70, 75\}$, attacked (resized and attacked with the anti-forensic scheme in [6]) and then re-compressed with $QF2 = 85$.[2] Then, we added these images to the double compressed images as further examples of the $H_1$ class. The number of images used for the manipulated class, then, raised to 16600 (7000 double + 4800 resized and 4800 CF attacked). Table 4.2 shows the results of the test against the SVM trained in such a way. The AUC is above 90% for almost all the processing operations and the counter-forensic attacks, thus confirming the good generalization capability of the detector. The good performance against non-aligned cropping shows that the detector is also robust to non-aligned DJPEG compression (or similarly, to a grid de-calibration attack), since non-aligned cropping between two compressions is equivalent to a non-aligned DJPEG.

### 4.4.3 Refined MPA-aware detector

From the results in Table 4.2, we see that when a geometrical operation like resizing and rotation is performed by using a nearest neighbor interpolation, the performance of the classification degrade. Then, we refined the MPA-aware detector by adding also some examples of such kind of manipulation in the $H_1$ class (thus adding further 4800 attacked samples). Table 4.3 shows the results of the refined detector. As expected, the performance with respect to resizing with NN interpolation improves. The performance with respect to rotation with NN interpolation also improves. Finally, the performance with respect to the other processing and attacks remain good.

[2] Notice that we did not consider the pair $(80 - 85)$ for the training, as we found experimentally that including attacked images with such a small difference between the $QF$s slightly reduces the performance of the classifier.

Table 4.2: AUC values of the MPA-aware classifier.

| **Attack** | DJPEG | 1st order MPA | Stamm Attack | wavelet denoise | median filtering | copyMove 256x256 |
|---|---|---|---|---|---|---|
| **AUC** | 0.99 | 0.98 | 0.96 | 0.90 | 0.98 | 0.99 |
| **Attack** | CLAHE | resize BIC0.9 | resize BIL0.9 | resize NN0.9 | rotation BIC5 | rotation NN5 |
| **AUC** | 0.92 | 0.92 | 0.95 | 0.80 | 0.91 | 0.81 |
| **Attack** | zoom 1.2 | crop align | crop no align | mirror | blur | seam-carving |
| **AUC** | 0.97 | 0.92 | 0.92 | 0.99 | 0.98 | 0.95 |

Table 4.3: AUC values of the refined MPA-aware classifier.

| **Attack** | DJPEG | 1st order MPA | Stamm Attack | wavelet denoise | median filtering | copyMove 256x256 |
|---|---|---|---|---|---|---|
| **AUC** | 0.99 | 0.98 | 0.96 | 0.91 | 0.98 | 0.99 |
| **Attack** | CLAHE | resize BIC0.9 | resize BIL0.9 | resize NN0.9 | rotation BIC5 | rotation NN5 |
| **AUC** | 0.91 | 0.92 | 0.94 | 0.92 | 0.92 | 0.91 |
| **Attack** | zoom 1.2 | crop align | crop no align | mirror | blur | seam-carving |
| **AUC** | 0.97 | 0.92 | 0.93 | 0.99 | 0.98 | 0.95 |

In order to get more insight into the impact that the $QF$s have on the performance, Table 4.4 and 4.5 show the AUC of the refined MPA-aware detector for the $QF$ pairs $(65, 85)$ and $(80, 85)$ respectively. Not surprisingly, the $(80, 85)$ case leads to worse results.

To verify that the classification results are not affected by the size of the images, Table 4.6 shows the results we obtained by testing the detector on the larger versions of the images of size $2144 \times 1424$. A well known problem with forensic tools based on machine learning, is that they may be affected by the problem of database mis-match [112], that is, the classifier has poor performance when tested with images coming from a different dataset with

Table 4.4: AUC of the refined MPA-aware classifier for the pair $(65, 85)$.

| **Attack** | DJPEG | 1st order MPA | Stamm Attack | wavelet denoise | median filtering | copyMove 256x256 |
|---|---|---|---|---|---|---|
| **AUC** | 0.99 | 0.99 | 0.98 | 0.96 | 0.99 | 0.99 |
| **Attack** | CLAHE | resize BIC0.9 | resize BIL0.9 | resize NN0.9 | rotation BIC5 | rotation NN5 |
| **AUC** | 0.96 | 0.96 | 0.97 | 0.95 | 0.97 | 0.95 |
| **Attack** | zoom 1.2 | crop align | crop no align | mirror | blur | seam-carving |
| **AUC** | 0.98 | 0.97 | 0.96 | 0.99 | 0.99 | 0.95 |

Table 4.5: AUC of the refined MPA-aware classifier for the pair $(80, 85)$.

| **Attack** | DJPEG | 1st order MPA | Stamm Attack | wavelet denoise | median filtering | copyMove 256x256 |
|---|---|---|---|---|---|---|
| **AUC** | 0.99 | 0.98 | 0.92 | 0.79 | 0.94 | 0.96 |
| **Attack** | CLAHE | resize BIC0.9 | resize BIL0.9 | resize NN0.9 | rotation BIC5 | rotation NN5 |
| **AUC** | 0.87 | 0.83 | 0.87 | 0.78 | 0.86 | 0.86 |
| **Attack** | zoom 1.2 | crop align | crop no align | mirror | blur | seam-carving |
| **AUC** | 0.89 | 0.85 | 0.84 | 0.99 | 0.91 | 0.93 |

respect to training. To verify that our system is not affected by this problem, we tested the refined MPA-aware classifier trained on RAISE-2K dataset using images from the Dresden database. Starting from the 752 uncompressed images made available in that database, with size 1936 × 1296, we generated single compressed, double compressed and attacked images according to the same procedure described so far. Table 4.7 shows the performance of the classifier when tested on a mix of images taken from both RAISE-2K and Dresden datasets for the most dangerous attacks among those considered in Table 4.3. As we can see the results do not differ much from those obtained

Table 4.6: AUC values of the refined classifier for larger images.

| **Attack** | DJPEG | 1st order MPA | Stamm Attack | wavelet denoise | median filtering | copyMove 256x256 |
|---|---|---|---|---|---|---|
| **AUC** | 0.99 | 0.98 | 0.96 | 0.91 | 0.90 | 0.98 |
| **Attack** | CLAHE | resize BIC0.9 | resize BIL0.9 | resize NN0.9 | rotation BIC5 | rotation NN5 |
| **AUC** | 0.91 | 0.91 | 0.93 | 0.93 | 0.91 | 0.90 |
| **Attack** | zoom 1.2 | crop align | crop no align | mirror | blur | seam-carving |
| **AUC** | 0.97 | 0.93 | 0.93 | 0.99 | 0.99 | 0.95 |

Table 4.7: AUC of the refined MPA-aware classifier tested on a mixture of imaged from RAISE-2K ($\approx 30\%$) and Dresden ($\approx 70\%$).

| **Attack** | DJPEG | wavelet denoise | resize BIC0.9 | resize NN0.9 | rotation BIC5 | rotation NN5 |
|---|---|---|---|---|---|---|
| **AUC** | 0.92 | 0.87 | 0.91 | 0.89 | 0.92 | 0.89 |

using images from RAISE-2K dataset only. The situation is similar when 100% of the images come from a different database, as we can see from Table 4.8, where we report the AUC of the refined MPA-aware classifier tested on Dresden and VISION datasets.

To show that results are not affected much by the choice of $QF2$, we also run experiments for a quite different value of the second quality factor, that is $QF2 = 95$. To train the model for $QF2 = 95$, the images of the training set $\mathcal{S}_{tr}$ were compressed with $QF1 = \{75, 80, 85, 90\}$, attacked (resized and attacked with the anti-forensic scheme in [6]) and then re-compressed with $QF2 = 95$. We did not consider $QF1$ lower than 75 during training, since they correspond to the easier cases for the detection and then including them in the training set does not help much. Table 4.9 reports the AUC results averaged on the test set built with $QF1 = \{70, 75, 80, 85, 90\}$, for some of the main attacks and processing.

Table 4.8: AUC of the refined MPA-aware classifier under mismatched dataset. The Dresden (first row) and Vision (second row) dataset are considered.

| **Attack** | DJPEG | wavelet denoise | resize BIC0.9 | resize NN0.9 | rotation BIC5 | rotation NN5 |
|---|---|---|---|---|---|---|
| **AUC (Dresden)** | 0.86 | 0.92 | 0.97 | 0.90 | 0.97 | 0.89 |
| **AUC (Vision)** | 0.87 | 0.89 | 0.94 | 0.81 | 0.95 | 0.81 |

Table 4.9: AUC of the refined MPA-aware classifier trained for $QF2 = 95$.

| **Attack** | DJPEG | Stamm Attack | wavelet denoise | median filtering | copyMove 256x256 | CLAHE |
|---|---|---|---|---|---|---|
| **AUC** | 0.99 | 0.95 | 0.95 | 0.97 | 0.97 | 0.88 |
| **Attack** | resize BIC0.9 | rotation BIC5 | zoom 1.2 | crop align | crop no align | seam-carving |
| **AUC** | 0.94 | 0.94 | 0.88 | 0.79 | 0.85 | 0.91 |

## 4.5 Final Remarks

In this chapter, we presented an adversary-aware double JPEG detector capable to work even in the presence of heterogeneous processing and CF attacks. We have conducted our tests in a rather controlled scenario: training and testing with the same $QF2$ and $QF1 < QF2$. As to the value of $QF2$, the performance may decrease in case of mis-match between training and testing. However, $QF2$ is generally known to the defender (since it can be derived from the JPEG bitstream or reliably estimated from the image), then many versions of the detector can be trained and used for different values of $QF2$. Regarding $QF1$, performance degrade when the detector is tested with $QF1 > QF2$. To get good performance in this case, examples of images for which $QF1 > QF2$ must be included in the training set, even in this way of the overall accuracy may decrease a bit (about 2% of the AUC value according to some preliminary tests we carried out[3]). In practice, better generalization capabilities can be achieved at the price of a reduction of performance. It is worth observing that the proposed approach is not meant for localization, and the performance are expected to decrease on small images (or patches). The average performance of the classifier for DJPEG detection trained with $QF2 = 85$, and 95 on smaller images (under no attack) are reported in Table 4.10.

Table 4.10: AUC of the refined MPA-aware classifier tested on small patches.

| Patches size | 536 × 356 | 268 × 178 |
|---|---|---|
| AUC ($QF2 = 85$) | 0.82 | 0.78 |
| AUC ($QF2 = 90$) | 0.89 | 0.80 |

[3] The results are referred to the MPA-aware detector in Section 4.4.2

# Chapter 5

# Detection of Image Contrast Manipulation Robust Against JPEG Compression

*"Forensic science offers great potential, as it draws on almost every discipline and, in doing so, creates widespread opportunity for innovation."*
Marl Walport

*"We can all see, but can you observe?"*
A.D. Garrett

JPEG compression is a powerful tool to remove manipulation traces, since quantisation can effectively reduce the distinctiveness of certain features without raising suspicion and without significantly degrading the perceptual quality of the image. In particular, a task which is known to be very challenging in the presence of JPEG post processing, is the detection of contrast adjustment. While in many case JPEG post processing is applied innocently (as JPEG is the most common image format), in many others it may be purposely applied by an attacker to erase the traces of manipulation. Then, designing a JPEG-robust contrast manipulation detector is of primary importance. This is the purpose of the research described in this chapter.

Specifically, in Section 5.2, we propose a solution based on JPEG-aware training and SVM classification, by limiting the analysis to the case of Adaptive Histogram Equalization (AHE). Then, in Section 5.3, we investigate the use of CNN-based classification for the same purpose, again coupled with JPEG-aware training, by focusing on the detection of a generic contrast adjustment. An introduction to the problem of contrast adjustment detection in the presence of JPEG and the related state-of-the-art is provided in Section 5.1.

## 5.1 An Overview on Contrast Adjustment Detection in the presence of JPEG post-processing

When creating a forgery, adjustment of the contrast and lighting conditions of image subparts is often performed to make the forgery more credible. Therefore, the detection of this kind of manipulation has been widely studied in image forensics. Due to peculiar traces left by contrast adjustment operators in the image histogram, early works were based on the analysis of image first order statistics [36, 65, 159]. Anti-forensic methods against forensic techniques based on first-order statistics have been developed as well; in addition to targeted approaches, aiming at removing the specific histogram artifacts the attacked detectors look at [90], universal approaches against generic histogram-based detectors have also been developed with good results [90]. Expectedly, it is quite easy to cope with such attacks by developing detectors based on second-order statistics [50,138]. Such ad-hoc detectors, however, fail when different attacks are considered. Moreover, since, in real applications, the attack is not known in advance, targeted anti-counter-forensic methods are simple of limited applicability. Moreover, in most cases, the attack consists in the application of one (or several) post-processing operations, e.g., a geometric transformation, filtering or compression (see Chapter 3, Section 3.2). Laundering attacks have in fact been shown to be very powerful against manipulation detectors [66].

The performance of contrast manipulation detectors proposed in the literature have been shown to decrease significantly in the presence of even mild post-processing and, in particular, all of them exhibit a poor robustness against JPEG compression [36, 65, 66, 140, 156]. Detection of contrast manipulation is not exception, and given poor resilience to JPEG compression is a common problem of detectors of contrast manipulation.

## 5.2 SVM-based Detector of Contrast Enhancement robust to JPEG

In this section, we present a system that is able to detect contrast enhancement by means of Adaptive Histogram Equalization (AHE) in the presence

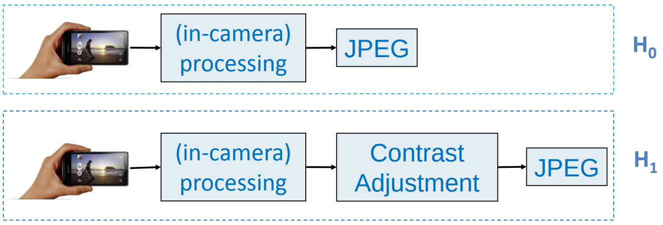


Figure 5.1: Detection task considered in Section 5.2 and Section 5.3.

of JPEG compression, by training a JPEG-aware SVM detector based on color SPAM features, i.e., an SVM detector trained on contrast-enhanced-then-JPEG-compressed images.

In the following, we first formalize the detection problem, then we describe the choice of the feature set and present the architecture of the proposed detector (based on a pool of Support Vector Machines (SVMs) trained in an adversary-aware modality).

The detection task we are facing with is schematised in Figure 5.1. We let hypothesis $H_0$ correspond to the case of pristine images and $H_1$ to the case of contrast adjusted (enhanced) images. In both cases, the images are JPEG compressed at the end with a given Quality Factor ($QF$). In this scheme, JPEG compression can be regarded to either as a post-processing or as a counter-forensic, laundering-type, attack.

In this section we focus on Adaptive Histogram Equalization (AHE) [160], which applies contrast enhancement on a local basis. The detection of such operator has not gathered much attention in the forensic literature. Compared to the detection of other global contrast enhancement operators (like for instance gamma correction and histogram stretching), the detection of AHE is more challenging since AHE does not introduce easily identifiable artifacts in the image histogram, even if it leaves traces in different domains that can be captured by CNNs (as shown in Section 5.3).

### 5.2.1 The CSPAM feature set

The choice of the feature set is a very important step in the design of our detection. In particular, we need to select a sufficiently large number features which are capable of capturing as many types of dependencies among neighboring pixels as possible. On the other hand, we want to limit the dimensionality of the feature set, so to be able to train an SVM. Common residual-based features (e.g., [59,69]), largely used in multimedia forensics for a wide variety of tasks (see Section 2.4), are designed for grayscale images and cannot be directly applied to color images. In this case, a possibility would be to extract the features from the luminance channel; however, contrast enhancement also modifies the relationship among color channels, and hence considering the luminance only (or, similarly, converting the images to grayscale), would result in a loss of possibly useful information. In order to take into account the relationships among color channels, we considered the color rich model features proposed in [60] for steganalysis (CSRMQ1), and adapted it to our case. Basically, the rich color feature space proposed in [60] consists of two different components. The first component is derived from the spatial rich model as in [59] (SRMQ1): specifically, the SRMQ1 features are computed for each color channel and added to keep the same dimensionality of grayscale images. The second component is a collection of 3-D color co-occurrences , computed from the same noise residuals as for the SRMQ1 model but formed across the three channels of each pixel. As described in the previous chapter, in its complete form, the SRMQ1 model considers many different types of residuals (39) and then the final feature space has a very large dimensionality (consisting of 12.753 features), which cannot be adopted for standard detection based on a single classifier. Therefore, in our case, we adopted a new feature set by using the SPAM (Subtractive Pixel Adjacency Matrix) feature set (described in Section 2.4.1.2), designed for one-channel images, as the base set for the color model. Specifically, according to the SPAM model, the first component of the feature vector is obtained by considering the second-order spatial co-occurrences (i.e., $d = 2$) of the first order residuals, with a truncation parameter $T = 3$, computed for each channel $R$, $G$, and $B$ separately, and then merged. For the second component, the (second-order) residual co-occurrences are computed with respect to the three channels, i.e. across the color bands ($R$, $G$ and $B$). We call CSPAM this simplification of the CSRMQ1 set. Since the dimensionality of the SPAM set is 686, the final dimensionality of the CSPAM set is $2 \times 686 = 1372$. Figure 5.2 illustrates the

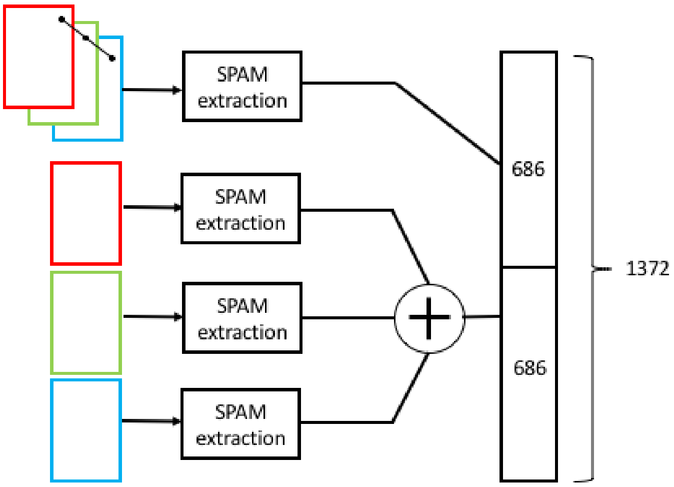


Figure 5.2: Illustration of the procedure for the extraction of the CSPAM feature set.

procedure that we followed for the extraction of the color feature set used for the SVM-based detection problem.

### 5.2.2 JPEG-aware detection scheme

Similarly to what happens with the contrast enhancement detectors proposed in the literature, if we train the SVM classifier based on the CSPAM feature set on pristine and enhanced images, without taking into account the JPEG compression in the end, the detector can correctly reveal the enhancement in the ideal scenario in which the enhancement of the contrast is the last step of the manipulation chain, but it completely fails in the presence of JPEG post-processing, even when JPEG compression is very mild.

To design a contrast enhancement detector robust to JPEG laundering attack, we trained several adversary-aware versions of the SVM classifier, where the classifier is trained with JPEG compressed images on one hand ($H_0$) and images subject to contrast enhancement followed by JPEG compression with different $QF$s on the other hand ($H_1$).

The overall architecture of the detector is reported in Figure 5.3. For a given JPEG image, the value of the $QF$ used for JPEG compression can be

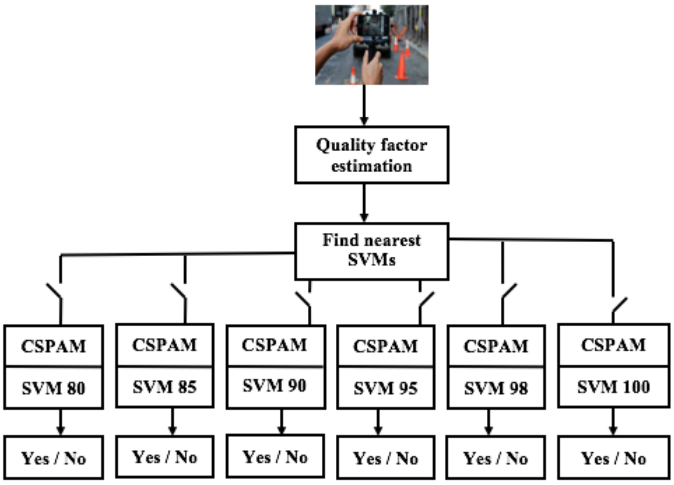


Figure 5.3: Scheme of the proposed JPEG-aware detector.

easily extracted (read) from the quantization table provided in the header of the image bitstream and used to select the most suitable version of the SVM classifier (i.e. the one trained with the $QF$ which is most similar to the one used to compress the test image).

In principle, we should train an SVM for every value of $QF$ and then use the corresponding SVM model for testing[1]. However, we experimentally verified that similar results can be obtained by training a lower number of SVMs for some selected values of $QF$ and then using the SVM corresponding to the closest $QF$. By referring to Figure 5.4, we see that the performance decay rather slowly (in terms of AUC) when the $QF$ of the test image departs from the $QF$ used for training (matched value). Notice also that, not surprisingly, the performance in the matched case increases for larger values of $QF$, since a weaker compression is less effective in erasing the traces of AHE. Based on our tests, we argued that a quantization step equal to 5 for medium-high $QF$ values, and 2 for very high $QF$s is appropriate. Then, we built our classifier by considering the six SVM models reported in the scheme of Figure 5.3.

Figure 5.5 shows the results obtained by training one SVM model with a

[1] Since we are interested in medium-high qualities, we take $QF > 80$ (much lower $QF$ are not very common in practice since the visual image quality degrades too much).

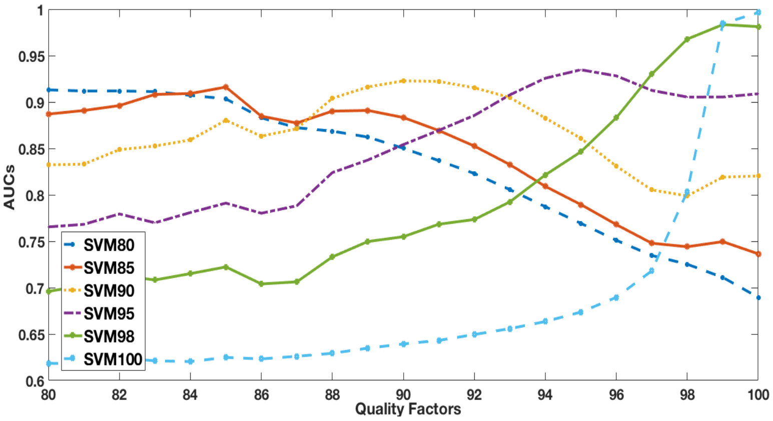


Figure 5.4: Performance of the SVMs as a function of the $QF$.

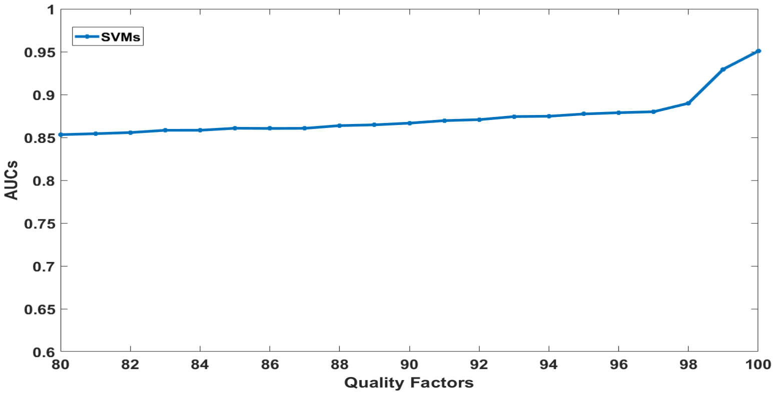


Figure 5.5: Performance of the SVM trained on the mixture of 6 $QF$s, as a function of $QF$.

mixture of the six $QFs$ (instead of 6 different SVM models). As we can see, the average performance are lower (average AUC = 0.89).

#### 5.2.2.1 Idempotency-based QF estimation

The proposed detector can be applied to JPEG images, in which case the $QF$ used to compress then can be obtained from the JPEG bitstream. However, if the compressed image is re-saved in uncompressed format (e.g., png, or bitmap) the proposed detector does not work any more. As we said, the attacker himself might re-save the image to hide the compression $QF$ (as a countermeasure to the JPEG-aware detection). Therefore, we also propose an algorithm to estimate the $QF$ directly from image pixels, thus extending the applicability of our detector. Another possibility to design a system which is robust to attacks would be to train a single SVM classifier for all or some selected values of the $QF$ (properly quantized). However, based on our experiments, by following this approach we get lower performance. Even if the drop is not serious, the reduction of the performance in this case makes us prefer the pool of SVMs, to the single SVM trained on a mixture of $QF$s. In order to get an estimate of the $QF$ from the pixel domain, we exploit the fact that JPEG compression is an (almost) *idempotent* operator, that is, whenever applied multiple times with the same $QF$, it produces the same result obtained with a single application. A similar property has been exploited in [161] for video codec identification to estimate $QF$.

Based on this observation, our algorithm works as follow:

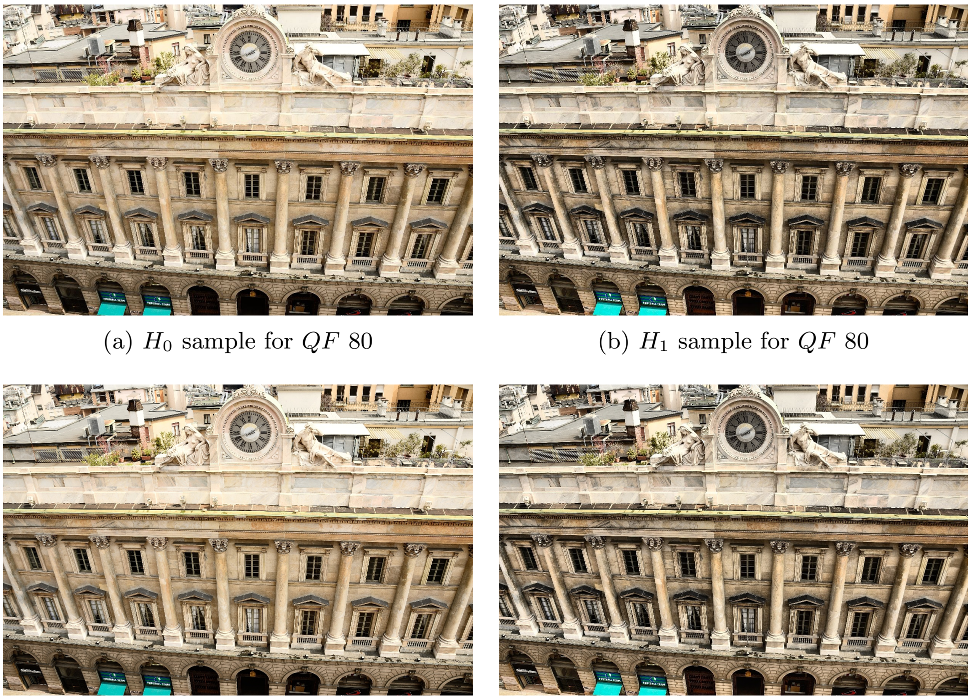
- the image under analysis $I$ is compressed with various $QF$s. We start from $QF = 50$ assuming that the image is never compressed with a lower $QF$. Let $I_{QF}$ denote the image compressed with quality factor $QF$;
- the $L1$ distance between the images before and after compression is computed for every $QF$, is $||I - I_{QF}||$
- the value of the $QF$ leading to the minimum distance is selected, that is $\hat{QF} = \arg\min_{QF} ||I - I_{QF}||$

To avoid the problem that QF tend to always produce small distances, we identified a critical $QF$ value, say $QF^*$. Then, the above algorithm is applied by considering $QF \in [50 : 1 : QF^*]$, that is, the local minimum is searched in $[50 : 1 : QF^*]$. If no local minimum is found inside this range, then a finer search is performed over the $QF$s larger than $QF^*$, and up to 98. For higher $QF$s, namely 99 and 100, the system guess is always 98. The reason is that very high values of $QF$ (99 and 100) are difficult to estimate with no errors and the detection performance of our system in these cases are generally very good even in presence of a mismatch between the real $QF$ and the one considered to train the SVM to training (see Figure 5.4). As a consequence, when $QF$ estimation is performed in the pixel domain with the idempotency-based algorithm, the SVM model for $QF$=100 (SVM100 in Figure 5.3) is never selected by the detector. The critical value $QF^*$ is experimentally set to 93.

### 5.2.3 Experimental Methodology

To produce the datasets for our experiments, we started from color images in uncompressed format, part of which were used for training and part for testing. The images for the $H_0$ and $H_1$ classes were built as detailed in Figure 5.1. The images were JPEG compressed with quality factor $QF$ to produce the $H_0$ samples, while, the images for the $H_1$ class were generated by first applying the AHE operator and then compressing them with quality factor $QF$ (the same as for $H_0$). For the unaware case, the training images were built according to the same scheme but without considering the JPEG compression stage at the end.

The Contrast-Limited implementation of AHE (CLAHE) was used for contrast enhancement [155]. With respect to ordinary AHE, CLAHE prevents the overamplification of noise in relatively homogeneous regions. This is done by clipping the histogram at a predefined value before computing the cumulative distribution function (CDF); this limits the slope of the transformation function (given by the CDF) which determines the contrast amplification. The value at which the histogram is clipped, called clip limit, depends on the normalization of the histogram and thereby on the size of the neighborhood region, which by default is $8 \times 8$. On color images, the straightforward application of CLAHE to each channel separately unnaturally changes the color balance and produces visually unpleasant images. A common strategy to work around this problem is to convert the images from the RGB to the HSV color space and then applying CLAHE only to the luminance channel, namely the $V_{channel}$. Then, the image is converted back to the RGB domain. We then followed this strategy to produce the AHE manipulated images in our case. In our experiments, the clip limit parameter for CLAHE was set to 0.004. Some sample images for the hypotheses $H_0$ and $H_1$ are provided in Figure 5.6 for $QF$ equal to 80 and 98. Regarding $QF$ values, the images used for training (both under $H_0$ and $H_1$) were compressed with $QF \in \{80, 85, 90, 95, 98, 100\}$, whereas for the test images all the $QF$s in the range $[80, 100]$ were considered. The Matlab environment was used to process the images, to train and test the SVMs (with the LibSVM library package [162]) and run the idempotency-based $QF$ estimator. In our tests, we also considered the GIMP software (and also Photoshop) for compressing the test images, in order to assess the performance of the detector in the presence of a mismatch in the compression software. Each SVM classifier was fed with the 1372-dimensional features (CSPAM) extracted from the color images. A Gaussian kernel was adopted, and the kernel parameters were determined by 5-fold cross-validation. In the



(a) $H_0$ sample for $QF$ 80

(b) $H_1$ sample for $QF$ 80

(c) $H_0$ sample for $QF$ 98

(d) $H_1$ sample for $QF$ 98

Figure 5.6: Visual comparison between an $H_0$ and $H_1$ images for two different $QF$s.

unaware case, we trained the SVM with uncompressed pristine and contrast manipulated images. In the aware case, to build the pool of SVMs detectors depicted in Figure 5.3, we separately trained the six SVMs (namely SVM80, SVM85, SVM90, SVM95, SVM98 and SVM100) on the corresponding JPEG compressed versions of the images.

### 5.2.4 Results

We considered uncompressed (tiff) images taken from the RAISE-8K dataset, consisting of about 8000 camera-native images (see Section 2.4.3). Specifically, we built our dataset as follows: 6000 images were used for the training set (1000 of which were used for tuning the kernel parameters, i.e., for internal cross-validation) and 1997 images for the tests. To get a faster feature computation, the images were subsampled to a size of 1072× 770.

#### 5.2.4.1 The unaware case

In this section, we show the results of unaware classification. The unaware SVM can perfectly classify uncompressed pristine and manipulated images, and the Area Under Curve (AUC) of the ROC curve for the classification is 100%. We also run some tests in the presence of laundering attacks, that is when both the pristine and manipulated images are subject to post-processing operations. In particular, we considered a case of filtering (median filtering with window size $3 \times 3$) and geometric transformations (resize with scaling factor 0.9, rotation with an angle of 5 degrees). In all these cases, the performance only slightly decreases, and the AUC remains above 90%. This shows that the CSPAM feature set is discriminative enough for our classification task.

When JPEG laundering is considered, however, the detector fails to classify the images, thus showing that the JPEG compression is very harmful and confirming the necessity of developing a JPEG-robust detector. Table 5.1 shows the detection performance of the unaware SVM detection.

Table 5.1: AUC values of the unaware SVM classifier.

| $QF$ | 80 | 81 | 82 | 83 | 84 | 85 | 86 |
|---|---|---|---|---|---|---|---|
| **AUC** | 0.5441 | 0.5429 | 0.5415 | 0.5390 | 0.5378 | 0.5370 | 0.5331 |
| $QF$ | 87 | 88 | 89 | 90 | 91 | 92 | 93 |
| **AUC** | 0.5310 | 0.5287 | 0.5274 | 0.5242 | 0.5199 | 0.5184 | 0.5118 |
| $QF$ | 94 | 95 | 96 | 97 | 98 | 99 | 100 |
| **AUC** | 0.5081 | 0.5027 | 0.4955 | 0.4871 | 0.4754 | 0.4570 | 0.4507 |

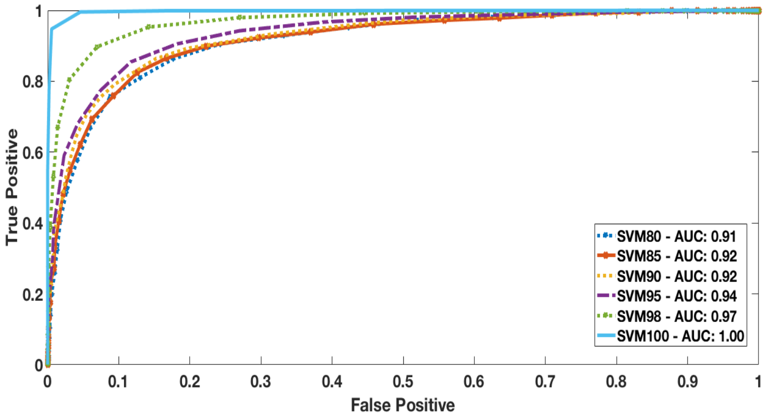


Figure 5.7: Performance of the aware SVMs for the classification task (under matched $QF$).

#### 5.2.4.2 Performance of the JPEG-aware detector

We now focus on the results achieved by the JPEG-aware detector. Figure 5.7 shows the performance of each SVM classifier tested under matched condition, that is, when the images considered for training and testing are compressed with the same $QF$. The performance improves significantly with respect to the unaware case. We observe that the performance reduce when the $QF$ decreases. This is expected since the lower the $QF$, the more the traces of AHE are erased by compression, and the detection task becomes harder. Arguably, for much lower quality factors, it is possible that the traces are almost completely erased; however, the quality of the images would also be

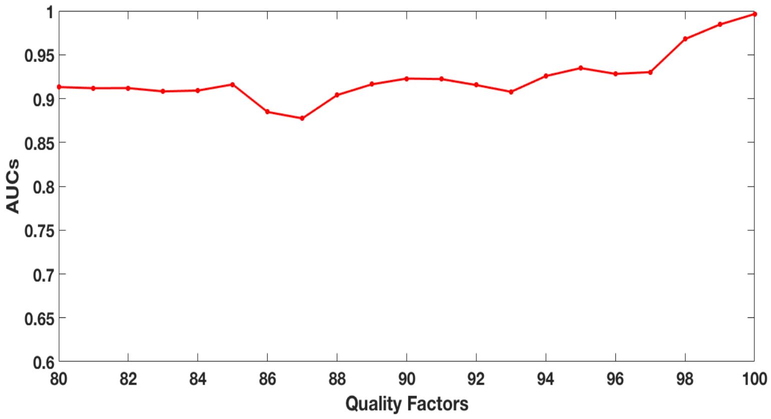


Figure 5.8: Performance of the system based on the pool of aware SVM classifiers.

seriously impaired.

In Figure 5.4 we report the results of the 6 SVMs for $QF \in [80,100]$. The performance of the system based on the pool of aware SVM classifiers when the $QF$ value is extracted from the JPEG header (i.e., perfect estimation) can be easily argued from these plots by considering for each image the closest $QF$ value in the set $\{80,85,90,95,98,100\}$ and then select the corresponding SVM for testing. The result is reported in Figure 5.8. From Figure 5.4, we also observe that using the minimum distance criterium for the SVM selection is a good choice. We also verified that by training only one SVM considering all the 6 $QF$ values above, the performance degrades significantly (the average AUC is 87%).

When the $QF$ is estimated on the pixel image according to the proposed idempotency-based approach, the performance is expected to decrease because of estimation errors. The average error in terms of L1 distance between real and estimated $QF$s under $H_0$ and $H_1$ is reported in Table 5.2 and Table 5.3 respectively. The average is computed on the 1997 images in the test set. The performance of the idempotency-based algorithm are pretty good, leading to an average estimation error below 0.1% for every $QF \leq 98$ Note that the average errors equal to 1 and 2 obtained, respectively, for $QF = 99$ and 100 are expected, given that for such $QF$s the algorithm always decide

Table 5.2: Average error of $QF$ estimation for $H_0$.

| $QF$ | 80 | 81 | 82 | 83 | 84 |
|---|---|---|---|---|---|
| **Average Error** | 0.0856 | 0.1002 | 0.0586 | 0.0676 | 0.0656 |
| $QF$ | 85 | 86 | 87 | 88 | 89 |
| **Average Error** | 0.0976 | 0.0190 | 5.0075e-04 | 0.0010 | 5.0075e-04 |
| $QF$ | 90 | 91 | 92 | 93 | 94 |
| **Average Error** | 5.0075e-04 | 0 | 0 | 0.1202 | 0.1022 |
| $QF$ | 95 | 96 | 97 | 98 | 99 |
| **Average Error** | 0.0315 | 0.0220 | 0.0015 | 0 | 1 |
| $QF$ | 100 | | | | |
| **Average Error** | 2 | | | | |

Table 5.3: Average error of $QF$ estimation for $H_1$.

| $QF$ | 80 | 81 | 82 | 83 | 84 |
|---|---|---|---|---|---|
| **Average Error** | 0.1017 | 0.1202 | 0.0686 | 0.0721 | 0.0631 |
| $QF$ | 85 | 86 | 87 | 88 | 89 |
| **Average Error** | 0.1107 | 0.0325 | 0.0015 | 0.0015 | 0.0010 |
| $QF$ | 90 | 91 | 92 | 93 | 94 |
| **Average Error** | 0.0010 | 0 | 0 | 0.1778 | 0.1402 |
| $QF$ | 95 | 96 | 97 | 98 | 99 |
| **Average Error** | 0.0451 | 0.0451 | 0.0015 | 0 | 1 |
| $QF$ | 100 | | | | |
| **Average Error** | 2 | | | | |

for 98 (see discussion in Section 5.2.2.1). Besides, we observe that the average error in the various cases is slightly larger under $H_1$ than under $H_0$. Figure 5.9 shows the performance of the system when $QF$ estimation is based on JPEG-idempotency. The performance reduction with respect to the case of perfect $QF$ estimation is pretty small (in the order of $10^{-3}$ on the average) and, expectedly (see the discussion in Section 5.2.2.1), pertains mainly to the case of higher $QF$ (99 and 100), where, however, the performance of the

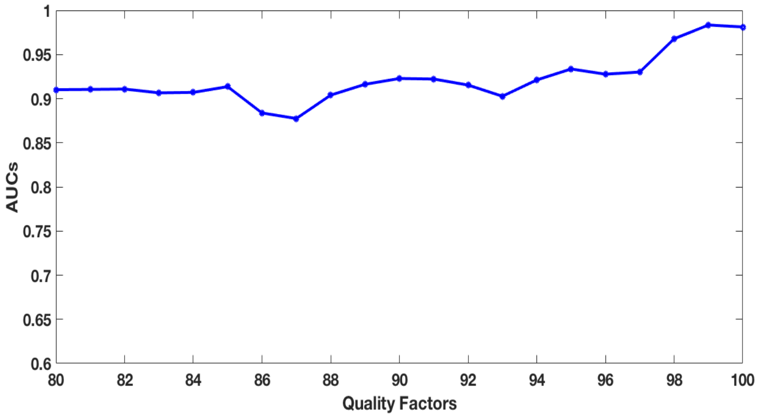


Figure 5.9: Performance of the system based on the pool of aware SVM classifier when the $QF$ is estimated by means of idempotency.

detector remains very good. We also verified that the performance for the case of uncompressed images remains good. In this case, as a result of the idempotency-based $QF$ estimation, the SVM98 is always selected. The AUC is 97,2% and then the performance reduction with respect to the unaware case is very small.

#### 5.2.4.3 Performance in the presence of software mismatch

All the results reported so far were obtained by working in the Matlab environment. However, the performance can be sensitive to a mismatch of the software used for compression, as different software may use different JPEG quantization tables. We may expect that this is especially the case when the idempotency-based $QF$ estimator is used; in this case, in fact, software mismatch might also lead to a wrong estimation. Then, we assessed the performance of the detector when the software used for compressing the test images is different from the one used to compress the images for training and inside the idempotency-based estimator. In particular, we used the GIMP software for JPEG compression of the test images. Then, the test images are compressed with GIMP while we keep using Matlab for the compression of the images used for training (and for the idempotency-based estimation). This corresponds to implement the last step of the processing chain in Figure

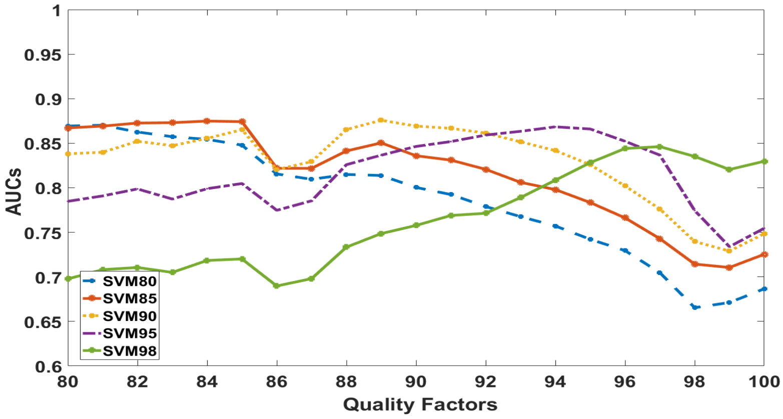


Figure 5.10: Performance of the SVMs as a function of the $QF$ in the presence of software mismatch. The test images are compressed with GIMP while the images used for training are compressed in Matlab.

5.1 with the GIMP software instead than with Matlab.

Figure 5.10 shows the results of the 5 SVMs for $QF \in [80{,}100]$. When the $QF$ value can be read from the JPEG header (i.e., perfect estimation), the performance of the overall system based on the pool of aware SVM classifiers can be easily derived from these plots. The performance reduces with respect to the matched case, although not drastically so (the AUC always remains above 80%). The average error in terms of L1 distance between real and estimated $QF$ is reported in Table 5.4 and Table 5.5 under $H_0$ and $H_1$ respectively. By comparing these results with those reported in the previous section for the matched case, we notice that the estimation remains good. Figure 5.11 shows the performance of the proposed detector in the presence of software mismatch when the $QF$ estimation is made in the pixel domain by means of the idempotency-based approach.

We also considered a mismatched case in which the Photoshop software is used (instead of GIMP) for compressing the test images. In this case, the idempotency-based approach has poorer performance with respect to the GIMP software case. By focusing on the Photoshop qualities 10, 11, 12 for the compression (which correspond to medium-high values of $QF$)[2], the per-

[2] In Photoshop, the strength of the compression is determined by setting a parameter for

Table 5.4: Average error of the $QF$ estimation for $H_0$ in the presence of software mismatch (test images compressed with GIMP).

| | | | | | |
|---|---|---|---|---|---|
| $QF$ | 80 | 81 | 82 | 83 | 84 |
| **Average Error** | 0.0200 | 0.0250 | 0.0130 | 0.0090 | 0.0100 |
| $QF$ | 85 | 86 | 87 | 88 | 89 |
| **Average Error** | 0.0195 | 5.0075e-04 | 0 | 0 | 0 |
| $QF$ | 90 | 91 | 92 | 93 | 94 |
| **Average Error** | 0 | 0 | 0 | 0.0401 | 0.0401 |
| $QF$ | 95 | 96 | 97 | 98 | 99 |
| **Average Error** | 0.0045 | 0.0030 | 0 | 0 | 1 |
| $QF$ | 100 | | | | |
| **Average Error** | 2 | | | | |

Table 5.5: Average error of the $QF$ estimation for $H_1$ in the presence of software mismatch (test images compressed with GIMP).

| | | | | | |
|---|---|---|---|---|---|
| $QF$ | 80 | 81 | 82 | 83 | 84 |
| **Average Error** | 0.0250 | 0.0300 | 0.0100 | 0.0090 | 5.0075e-04 |
| $QF$ | 85 | 86 | 87 | 88 | 89 |
| **Average Error** | 0.0260 | 5.0075e-04 | 0 | 0 | 0 |
| $QF$ | 90 | 91 | 92 | 93 | 94 |
| **Average Error** | 0 | 0 | 0 | 0.0376 | 0.0200 |
| $QF$ | 95 | 96 | 97 | 98 | 99 |
| **Average Error** | 0.0030 | 0.0030 | 0 | 0 | 1 |
| $QF$ | 100 | | | | |
| **Average Error** | 2 | | | | |

formance (AUC values) in the case of $QF$ estimated from the image pixels are 79, 85 and 90 respectively.

the image quality in a (non linear) scale from 0 to 12.

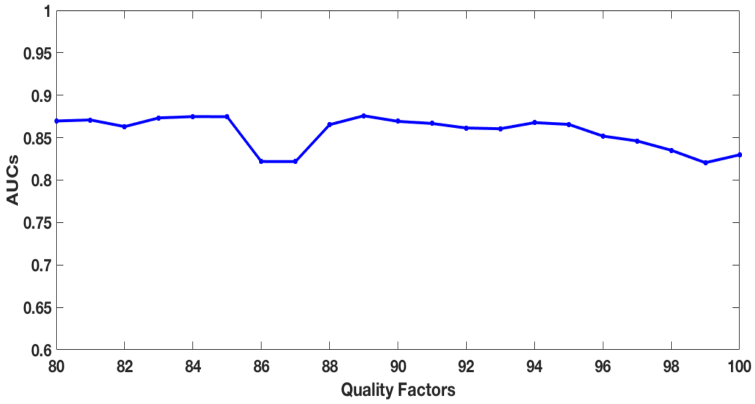


Figure 5.11: Performance of the detector in the presence of software mismatch (GIMP software).

## 5.3 CNN-Based Detection of Generic Contrast Adjustment robust to JPEG Compression

In this section, we describe a JPEG-robust detector of contrast adjustment by resorting to CNN-based classification. In particular, leveraging on the superior capability of CNNs, to authomatically learn discriminative features also for very complicated tasks, we look for a generic detector of contrast adjustment, that is, a detector that generalizes well to a wide variety of tonal adjustments. The proposed method relies on a single CNN. The CNN is directly fed with the image pixels (with no pre-processing), hence the discriminative features for our problem are self-learned by the CNN. More specifically, the proposed detector is based on a JPEG-aware, patch-based CNN, which is used to classify image regions, i.e. image patches. A test image is then divided into patches which are tested separately by feeding them to the CNN. The soft patch scores (CNN outputs) are collected and the global decision on the image is performed on the score vector (in this way, we indirectly exploit the fact that patches coming from a same image are generated under the same hypothesis, hence assuming that the image is either pristine or globally manipulated in contrast). All the compression $QF$s inside a range of values are considered to train the CNN.

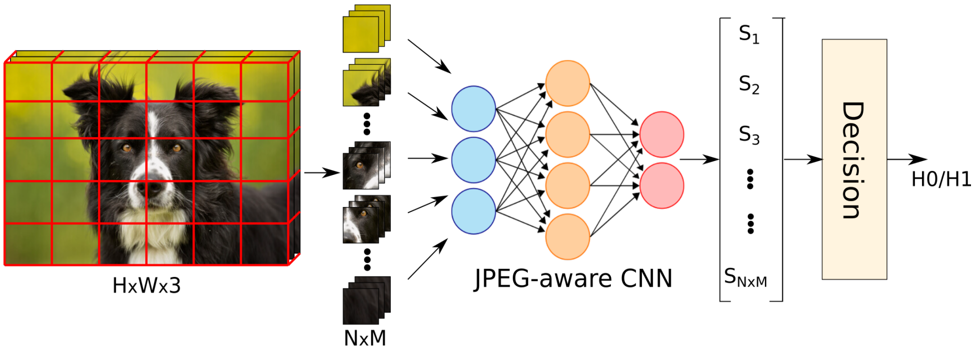


Figure 5.12: Pipeline of the proposed generic, JPEG-aware, contrast adjustment detector. Adaptive histogram equalization, gamma correction (both compression and expansion) and histogram stretching are used to train the network.

We point out that, in this case, the pooled structure, where several CNNs are specialized to work for one $QF$ each, is not more effective than a single CNN trained on all the $QF$s the performance depending rather on the size and composition of the training set.

### 5.3.1 Proposed System

The detection task we are facing with is the same illustrated in Figure 5.1, which the architecture of the proposed detection scheme is reported in Figure 5.12. The color image is divided into non-overlapping patches of size $64 \times 64$ which are fed to a JPEG-aware CNN detector. The patch scores, i.e. the CNN soft outputs for all the patches, are then collected and the final decision is based on the score vector $s = (s_1, s_2, ...s_{N \times M})$ (where $N \times M$ is the total number of blocks). The decision is made by simply thresholding the sum of the scores, i.e. according to the statistic $\sum_i s_i/(M \cdot N)$. Since patches coming from the same image are drawn under the same hypothesis, such normalised sum is expected to be large in one case (contrast adjusted image) and small in the other (pristine image)[3].

The JPEG-aware CNN is trained with JPEG compressed images on one

[3] We are considering the case of globally manipulated images.

hand ($H_0$) and images subject to contrast adjustment followed by JPEG compression on the other ($H_1$). The network architecture and the training strategy are detailed in the following sections. In the attempt to build a detector which generalizes to unseen adjustments, we consider contrast adjustments of different nature to train the network. Specifically, the processing we selected are: adaptive histogram equalization, gamma correction (both compression and expansion of the contrast) and histogram stretching.

Regarding the compression $QF$, we focus on values ranging from medium-high to high values (i.e., $QF \geq 80$), since these are the values commonly used in practical applications.

#### 5.3.1.1 CNN architecture

Our first attempts to train a network for our problem by using architectures similar to those adopted for other forensic tasks [8, 73, 78] were unsuccessful. A possible explanation is the following: while some processing operations, e.g. local filtering and double JPEG, introduce local patterns that a properly trained CNN with few layers is able to 'easily' learn, common contrast adjustments do not leave local visual artifacts, thus calling for the adoption of deeper models. We were in fact able to get higher accuracies by switching to deeper architectures, with small kernel sizes and small strides of the convolutional layers, inspired by those adopted in image classification and pattern recognition applications [163]. In particular, as suggested in [163], we adopted a kernel size of 3 × 3 and stride 1 for all the convolutional layers, and only 1 fully connected layer. We set the number of convolutional layers to 9, which, although lower than that adopted in [163] (16-19), is still a significant depth compared to those commonly considered for forensic tasks [8, 73, 78] (up to 4-5).

More specifically, the structure of our network for patch classification (see Figure 5.13) is detailed as follows: it takes a color patch of size 64 × 64 as input and consists of,

- **5 convolutional layers** followed by a **max-pooling** layer. In the first convolutional layer 32 filters are applied. Then, the number of filters increases by 32 at each layer. For all the filters, the kernel size is 3 × 3 and the stride is always 1. Max-pooling is applied with kernel size $2 \times 2$ and stride 2 producing a final $27 \times 27 \times 160$ feature map.

- **3 convolutional layers** followed by a **max-pooling** layer. As before, the number of filters of size $3 \times 3$ (applied with stride 1) increases by 32 at each layer. The pooling is the same as before, yielding a $10 \times 10 \times 256$ feature map.
- A final **convolutional layer** with 128 filters of size $1 \times 1$ generating a $10 \times 10 \times 128$ feature map.
- A **fully-connected layer** with 250 input neurons, dropout 0.5, and 2 output neurons, followed by a softmax layer (last 3 blocks in the scheme of Figure 5.13).

Some comments regarding the main features of the above architecture are in order: the use of many convolutions (5) before the first pooling layer permits to consider a large receptive field for each neuron, which is good to capture relationships among pixels in large neighborhoods; the stride 1 permits to retain as much spatial information as possible. The purpose of the final convolutional layer is to reduce the number of parameters by halving the number of maps (from 256 to 128), without affecting spatial information. The adoption of only one fully connected layer also permits to reduce the number of parameters without affecting too much the performance. Finally, we observe that using small patches ($64 \times 64$) permitted us to increase the depth of the network for the same number of parameters. The use of small patches is also suitable for tampering localization (the detection accuracy is then raised by aggregating the patch scores).

#### 5.3.1.2 CNN training strategy

We obtained the JPEG-aware CNN model by training the network in two steps. First, the network is trained to recognize between patches coming from pristine and contrast-adjusted images for the uncompressed case, getting a (unaware) pre-trained model. Then, the aware model is obtained by fine-tuning the unaware network, by feeding the CNN with JPEG compressed examples of the above classes. Since the network is pretty deep and then the

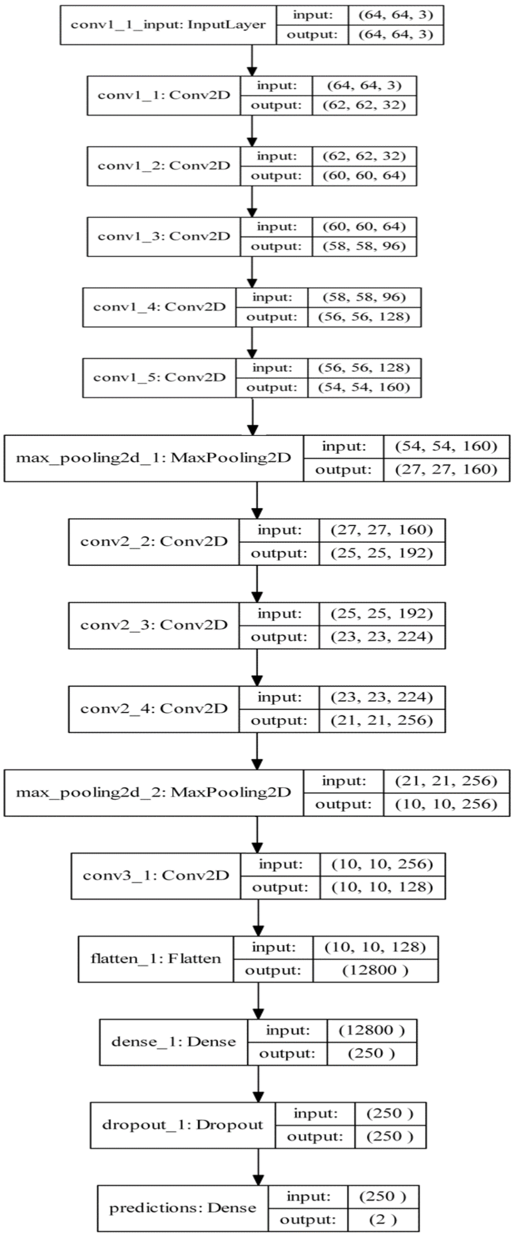


Figure 5.13: Architecture of the proposed network.

number of images used for training is very large, we performed compression on-the-fly by augmenting the data inside the network; hence, the compression is performed directly on the $64 \times 64$ patches (that is, after image splitting). Such a strategy is viable because the JPEG compression is a local operator which can be applied separately on multiples of $8 \times 8$ image patches producing the same result as if it were applied on the entire image.

### 5.3.2 Experimental Methodology

We built the training and testing sets by starting from color images in uncompressed format. The images for the $H_0$ and $H_1$ classes were produced as detailed in Figure 5.3. The adjustment of the contrast under $H_1$ is obtained by considering several algorithms. As we said, to generate the images used for training, we considered the following operators: Adaptive Histogram Equalization (in particular, the refined, Contrast Limited, implementation, CLAHE, already used for the SVM case), Gamma Correction ($\gamma$ `Corr`), and Histogram Stretching (HS). Such operators are designed for one-channel images; to make them work on color images, we applied them as follows: for the images processed with CLAHE, we followed the same steps described in Section 5.2.3, first converting the images from RGB to HSV, applying the enhancement to the luminance channel only ($V_{channel}$), and then converting back to the RGB domain The same strategy is adopted to generate the images processed with `HS`. Finally, for the $\gamma$ `Corr`, the contrast is modified by applying the operator to each channel (R, G and B) separately. The above operators are applied in equal percentage to generate the class of contrast adjusted images ($H_1$). Regarding the parameters, the clip-limit parameter for CLAHE is set to 0.005, the $\gamma$ value to 1.5 and 0.7 (randomly chosen with probability 0.5), and the saturation percentage of the HS to 5% for both black and white values. We verified that the above choices do not introduce visually unpleasant artifacts.

To generate the test images, we also considered different values of the parameters for the same operators (to assess the performance under parameter mismatch), and different operators, by processing the images with adjustment tools provided by Photoshop. In particular, we considered the following tonal adjustments:

- AutoContrast, AutoColor, and AutoTone; algorithms which operate differently with respect to the color channels. Clipping is set to 7% for AutoContrast and `AutoColor` and to 5% for AutoTone; the snap neutral midtones option is selected for the AutoColor;

- Curves_S; a (hand-made) smooth S-curve is applied to enhance the contrast in the midtones;

- Brightness, and Contrast; generic tools for enhancing and reducing brightness and contrast; for the enhancement, we set Brightness to 50 (Brightness+) and Contrast to 70 (Contrast+), while for the reduction, we set Brightness to -70 (Brightness-) and Contrast to -50 (Contrast-);

- Histogram Equalization (HistEq).

The HistEq manipulation is considered for completeness: although its visual impact is much stronger with respect to that of the other manipulations, and hence is rarely adopted in practice. The HistEq manipulation is often considered in multimedia forensic literature.

Regarding JPEG compression, we randomly selected the $QF$s (uniformly) in the range $[90, 100]$ to compress the images used for training. For testing, we also considered images compressed with $QF$ 85 and 80.

### 5.3.3 Results

A before, for these experiments, we considered uncompressed, camera-native, images (TIFF) taken from the RAISE-8K dataset, splitted into training and test set, and then contrast-adjusted to produce the images for $H_1$ in the unaware case (i.e., without the final JPEG). The images are then divided into $64 \times 64$ patches for CNN training and testing: $2 \times 10^6$ patches per class (coming from more than 1000 training images) were selected to train the CNN, whereas $2 \times 10^5$ patches were used for testing. In the aware case, the patches are JPEG compressed with $QF \in [90, 100]$. The overall performance of the detector are tested on 300 images from the test set, both uncompressed and compressed with $QF = \{100, 98, 95, 90, 85, 80\}$. The images used for training were all processed with the OpenCV library for Python. For the tests, the Photoshop software was adopted. We used the TensorFlow framework, via the Keras API [164], to implement the CNN. We ran our experiments using 2x Asus GeForce GTX1080TI - 11GB DDR5 gpu. The Adam solver is used with learning rate $1e-4$ and momentum 0.99. We set the batch size for training and testing to 32 images. Both the aware and unaware models were trained for 3 epochs.

When training and testing are performed with uncompressed images (unaware case), the average test accuracy of the CNN on image patches is 93.5%, where the average is taken on the 3 manipulations, i.e., CLAHE, $\gamma$ `Corr` (compression and expansion) and HS, and on all the $QF$s inside the training range. For the overall system, we get almost perfect classification, that is, the Area Under Curve (AUC) is 99, 8%, which is in line with the state of the art [66]. A noticeable strength is that here these performance are achieved by one (generic) system only, rather than using separate systems each specialized in one manipulation. By testing the unaware detector with JPEG compressed images, the performance drop to AUC = 56% thus showing that the CNN model is not robust to JPEG laundering attack.

Concerning the JPEG-aware case, the average accuracies that we obtained at the patch level in the range of $QF$s in $[90, 100]$ are: 0.84 for CLAHE, 0.72 for $\gamma$ `Corr` and 0.79 for `HS`. These accuracies are not very high; however, the performance are moderately good with respect to all the contrast adjustment operators. We also observe that specializing our network to work with one $QF$ only, we could have obtained higher performance at the patch level; however, as said before, to be robust against common manipulations (as recompression and saving in uncompressed formats), we look for a detector of generic contrast adjustments which works well on a range of $QF$s. The overall performance of the detector on full images are reported in Table 5.6 in terms of AUC, for both matched and mismatched processing parameters. The CLAHE manipulation is the easiest to detect (the AUC is always above 98%). The most difficult case corresponds to $\gamma$ `Corr`, where the AUC is below 90% for $QF \leq 95$. This behavior is due to the fact that thus kind of adjustment is difficult to detect by itself and mostly to the fact that the CNN is simultaneously trained with values smaller and larger than 1, corresponding to a compression and an expansion of the contrast[4]. Notably, these results

[4] We verified that if the detector is trained with $\gamma = 1.5$ only (gamma expansion), the

Table 5.6: Performance (AUC) of the detector under matched processing. The matched parameters are in bold.

| | | | *QF* | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | **no jpg** | **100** | **98** | **95** | **90** | **85** | **80** | **75** |
| **CLAHE** | 0.003 | 100 | 99.9 | 99.8 | 98.9 | 97.6 | 97.1 | 96.8 | 96 |
| | **0.005** | 100 | 99.9 | 99.9 | 99.4 | 98.9 | 98.8 | 98.5 | 98 |
| | 0.007 | 100 | 99.9 | 100 | 99.6 | 99.1 | 98.9 | 98.7 | 98.5 |
| $\gamma$ **Corr** | **1.5** | 98.8 | 98.5 | 94.2 | 89.2 | 87 | 84 | 81.2 | 81 |
| | 1.7 | 99.4 | 98.9 | 95.7 | 91.8 | 90.4 | 89.7 | 89.2 | 88.1 |
| | **0.7** | 99.1 | 97.1 | 92.3 | 87.3 | 85.6 | 81 | 78 | 69 |
| | 0.6 | 99.7 | 99.5 | 97.3 | 91.6 | 86.7 | 83.7 | 80.1 | 77.3 |
| **HS** (%) | 3 | 99.6 | 98.1 | 95.8 | 91.4 | 87.8 | 85.7 | 83.5 | 83 |
| | **5** | 99.5 | 98.9 | 97.6 | 93.7 | 92.6 | 91.5 | 90.3 | 89.4 |
| | 7 | 100 | 99.3 | 98.3 | 95.5 | 94 | 93.7 | 93.6 | 93 |

significantly improve those achieved by the SVM-based approaches from the literature, where, depending on the specific contrast adjustment considered (gamma correction and histogram stretching), the AUC may drop to about 60% [156], and 72% [66] on the average, in the presence of JPEG compression in the same range.

We observe that the performance are good in the presence of a mismatch in the processing parameters: better performance are obtained when the adjustment is stronger than in the matched case, and worse when it is weaker. The performance remain very good in the absence of JPEG: the AUC is 99.6% on the average in the matched case, which is in line with the AUC achieved by the unaware detector. Expectedly, performance decrease as $QF$ decreases. However, good robustness to JPEG compression is achieved (at least for CLAHE and HS) also when the $QF$ is 85 and 80, which are outside the training range, whereas, below 80, performance become poorer. It is worth observing that, for a fixed false alarm rate, the threshold on the aggregated score changes by

AUC for the $\gamma$ Corr is above 97% for every $QF \geq 85$. In this case, however, the performance with respect to a compression ($\gamma < 1$) are very poor even with large $QF$ (e.g. AUC = 78% for $QF = 95$).

Table 5.7: Performance (AUC) of the detector for different tonal adjustments.

$QF$

| | **no jpg** | **100** | **98** | **95** | **90** | **85** | **80** |
|---|---|---|---|---|---|---|---|
| HistEq | 100 | 99.9 | 99.9 | 99.5 | 98.3 | 96.9 | 94.8 |
| Brightness+ | 97.5 | 97.7 | 95.2 | 93.6 | 91.2 | 87.8 | 85.6 |
| Contrast+ | 99.1 | 100 | 99.6 | 97.9 | 94.7 | 91.9 | 87.1 |
| Brightness- | 96.7 | 97.3 | 93.3 | 90.1 | 84.2 | 78.8 | 75.6 |
| Contrast- | 98.8 | 99.6 | 96.4 | 91.2 | 87 | 82 | 80 |
| Curve_S | 99.6 | 99.8 | 99.8 | 99.1 | 97.7 | 96 | 93.6 |
| AutoContrast | 95.9 | 94.7 | 93 | 91.9 | 90.2 | 89 | 86.5 |
| AutoColor | 98.2 | 98.6 | 96.8 | 95.3 | 93.7 | 91.8 | 89.1 |
| AutoTone | 99.5 | 99.5 | 99 | 98.2 | 97.2 | 96.1 | 94.5 |

varying the $QF$: specifically, for a false alarm of 5%, the threshold ranges in $[0.56 : 0.71]$. Note that, since the last compression QF is always known (or it can be estimated), such a variability of the threshold is not a problem. Table 5.7 shows the results under various contrast/brightness adjustment performed with Photoshop. Based on these results, we can argue that the CNN-based detector scales well with respect to the adjustment type maintaining good performance when the tones of the image are adjusted in different ways and, possibly, selectively in different tonal ranges (Curve_S), and when the adjustment operates differently on the color channels (the Auto processing). The AUC is large with respect to all the $QF$s for some of the processing (AutoTone, Curve_S, HistEq) and, in general, it remains above 90% in most of the cases.

## Chapter 6

# Improving the Security of Image Manipulation Detection Through One-and-a-half-class Multiple Classification

*"If you spend more on coffee than on IT security, you will be hacked, What's more, you deserve to be hacked."*
White House Cybersecurity Advisor, Richard Clarke

Protecting image manipulation detectors versus perfect knowledge attacks requires the adoption of detector architectures, that are intrinsically difficult to attack. So, this chapter is devoted to design a multiple classifier architecture, referred to as a 1.5C classifier, to limit the damage made by an attacker with perfect knowledge acting against an image manipulation detector. The 1.5-Class (1.5C) architecture consists of one 2C classifier (2-Class classification), and two 1C classifiers (1-Class classification) run in parallel followed by a final 1C classifier.

This chapter is organised as follows: In the Section 6.1, we describe previous attempts to improve the security of image forensics analysis by resorting to 1C classification. Then, in the Section 6.2, we introduce the proposed 1.5C classifier for image manipulation detection and describe the strategy we followed to train it. In the Subsection 6.2.4, we describe the methodology we used to evaluate the effectiveness of the proposed system, while in Section 6.3, we present the corresponding experimental results. The chapter ends in the Section 6.4 where we summarise the main results of this research.

## 6.1 Prior art

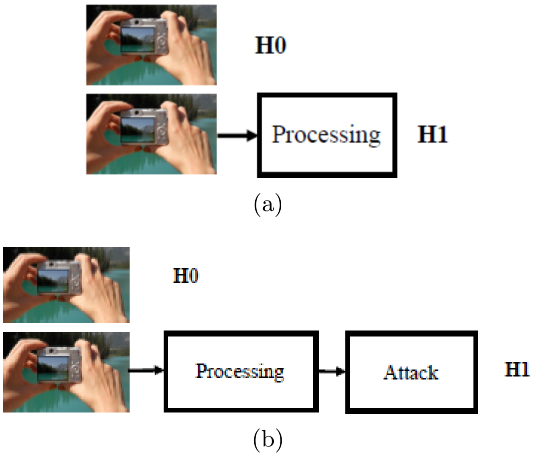
The use of 1C classifiers is not novel in multimedia forensics and security-related applications. In [165], the authors resort to 1C classification for video forgery detection, in the attempt to devise a tool which works under challenging conditions, like those encountered in social networks. In particular, the authors use of an architecture based on *autoencoders* trained on pristine data. When used in this way, autoencoders behave as 1C classifiers, with a large reconstruction error between the input and output being interpreted as an anomaly, i.e. a forgery. More in general, one class modelling is popular for anomaly detection in many different applications, where a good statistical characterisation under abnormal condition is not available. As an example, we mention the problem of acoustic novelty detection [166] or the problem of detection of abnormal events in video sequences [167]. A method for adversarial anomaly detection, which exploits the combination of multiple 1C classifiers to increase the hardness of evasion attacks against intrusion detection systems is provided in [168].

1C classification has often been proposed as an alternative solution to conventional multiclass algorithms, that classify a sample based on a number of pre-defined categories, when an exhaustive list of such categories does not exist [169]. This problem is often referred to as classification in *open set* conditions. Open set problems have been studied in several image forensic and security-oriented applications, such as fingerprint spoof detection [170] source device attribution [171], and camera model identification [172, 173]. In [173], in particular, a combination of one-class and multi-class SVMs is used to simultaneously recognize the camera model among in a known set and, at the same time, identify outliers, and models.

More recently, 1C classifiers have been used in conjunction with generative adversarial models (GANs) to design detectors which work under the assumption that very few or no instances of malicious samples are available for training. This is the case in [174], where the problem of forgery detection of satellite imagery is addressed, and in [175], with regard to general fraud detection, e.g. in reputation systems.

## 6.2 Proposed System

In this section, we first formalise the detection problem addressed in this chapter, then we describe the 1.5C architecture originally proposed in [97] for pattern recognition applications, the choice of the feature set, and the methodology that we followed to train the 1.5C classifier.



(a)

(b)

Figure 6.1: General manipulation detection task considered in this chapter (a) and its adversarial version (b).

The basic detection task addressed in this section is schematised in Figure 6.1(a). Hypothesis $H_0$ corresponds to the case of pristine images produced by an acquisition device without any subsequent processing. $H_1$ corresponds to the case of processed or manipulated images. Figure 6.1(b) depicts the case in which the same detection task is carried out in an adversarial setting, in the presence of an attacker aiming at impeding a correct detection. Specifically, we assume that the goal of the attacker is to avoid that the manipulation is detected, that is, the attacker's goal is to induce a missed detection error. In the sequel, we denote with $P_{md}$ the probability of a missed detection error, namely the probability that a manipulated image is detected as a pristine image (also indicated as $P(H_0|H_1)$), and with $P_{fa}$ the false alarm probability, that is the probability that a pristine image is detected as being manipulated

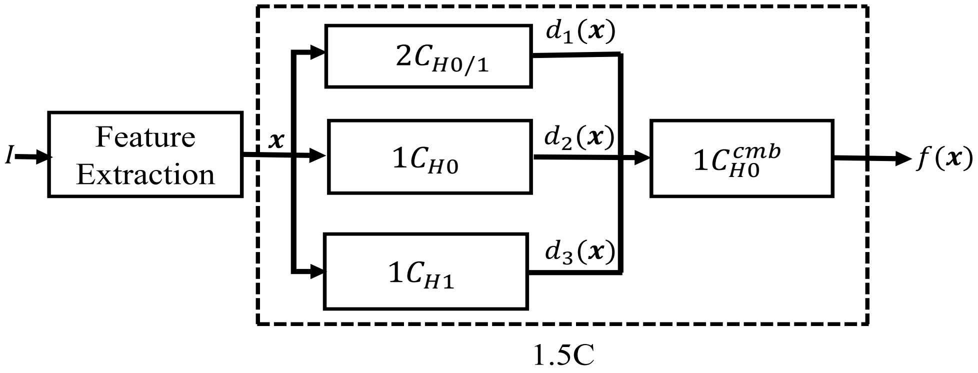


Figure 6.2: Architecture of the one-and-a-half class (1.5C) classifier. $I$ is the input image, while $\mathbf{x}$ is the feature vector extracted from $I$.

(namely, $P(H_1|H_0)$).

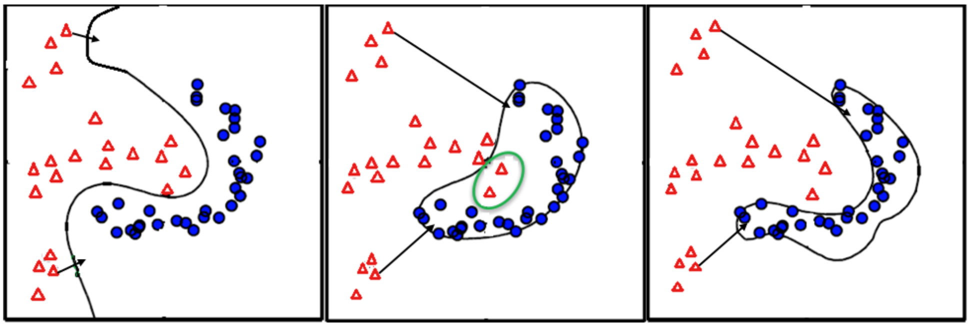
### 6.2.1 Architecture of the one-and-a-half-class (1.5C) classifier

The architecture of the one-and-a-half-class classifier adopted in this work is depicted in Figure 6.2. Given an image $I$, a feature vector $\mathbf{x}(I)$, or simply $\mathbf{x}$, is extracted and used to feed a multiple classifier whose first stage consists of the parallel application of three classifiers: a 2C classifier, trained with examples belonging to both classes (in our case, pristine and manipulated images) and two 1C classifiers, one trained with images belonging to the $H_0$ class (pristine images) and the other trained with images from the $H_1$ class (manipulated images). The outputs of these classifiers represent the input of a final 1C classifier, referred to as combination classifier. Since the goal of our work is to increase the security in the presence of integrity violation attacks, the final classifier is trained with pristine data.

We denote with $d_1(\mathbf{x}), d_2(\mathbf{x})$ and $d_3(\mathbf{x})$ the output respectively of the 2C, 1C trained on pristine images and 1C trained on manipulated images classifiers; $f(\mathbf{x})$ denotes the decision function of the downstream 1C combination classifier trained with pristine images. For ease of reference, in the sequel we indicate the above classifiers with the acronyms $2C_{H_{0/1}}, 1C_{H_0}, 1C_{H_1}$ and $1C_{H_0}^{cmb}$, as indicated in Figure 6.2.

The rational idea behind the 1.5C architectureis the following: 2C classifiers are often capable of achieving high accuracy in the absence of attacks, however they do not generalize well to examples that were not properly represented in the training phase, thus easing the task of the adversary, that can exploit the presence of unexplored regions of the features space to carry out his attack. Such a situation is exemplified in part (a) of Figure 6.3, where the attacker exploits the presence of empty regions to induce a missed detection error (for simplicity, the figure refers to a case of perfect classification in the absence of attacks). As a result, the 2C classifier may be easily attacked as shown for instance in [106,133] in the case of image manipulation and forgery detection. On the other hand, by defining a closed region enclosing the samples from one class only - usually the $H_0$ class, 1C classifiers are intrinsically more robust against attacks, even if they may get worse performance in the absence of attacks. This behavior is illustrated in Figure 6.3 (b), where we see that moving a sample from the $H_1$ to the $H_0$ region requires a larger distortion, due to the closeness of the acceptance region for $H_0$. Such an advantage comes at the price of a reduced accuracy in the absence of attacks, since the one class classifier is not able to properly shape the closed acceptance region given that it is trained only on examples generated under $H_0$ (the samples for which a missed detection error occurs are highlighted by a circle in the Figure).

The goal of the 1.5C classifier is to simultaneously retain the advantages of 2C and 1C classification, as illustrated in Figure 6.3 (c). In particular, the 1.5C classifier is expected to have a similar robustness against attacks as the 1C classifier (moving a sample from $H_1$ to $H_0$ requires a similar distortion), while the acceptance region is better shaped with respect to the 1C classification case, then the performance in the absence of attacks are improved compared to the 1C classifier, and are more similar to those obtained by the 2C classifier (the same perfect classification is achieved in the illustrative example in the figure). This behaviour is confirmed in our experiments for all the manipulation detection tasks we have considered (see the results in Section 6.3.2 and 6.3.3).



(a) 2-C (b) 1-C (c) 1.5C

Figure 6.3: Pictorial representation of the decision regions for 2C, 1C and 1.5C classifiers. Blue dots refer to $H_0$ hypothesis, red triangles to $H_1$. The goal of the attacker is to take samples belonging to $H_1$ detection region and move them inside the region for which the detector decides in favour of $H_0$. The performance of the 1.5C classifier are in line with those of the 2C, while it is expected to be more robust under attacks, since a larger distortion, exemplified in the picture by a longer arrow, is needed to move a sample from the $H_1$ to the $H_0$ region. The presence of red triangles within the $H_0$ region in part b) illustrates the lower performance of the 1C classifier in the absence of attacks.

### 6.2.2 Implementation of 1.5C detector and choice of the feature set

The 1.5C architecture outlined in the previous section is a general one, and, in principle, can be implemented by resorting to any kind of 1C and 2C classifiers. We decided to implement all the classifiers the 1.5C classifier consists of by means of Support Vector Machines (SVMs). The main reason is the ease with which SVMs can be used to implement 1C classification. It goes without saying that the use of alternative architectures can be explored as a future research direction.

With regard to the choice of the feature set, we tried to balance between two opposite requirements. On one hand, the features have to be powerful enough to capture the different types of dependencies among neighbouring pixels in pristine and manipulated images. On the other hand, we want to keep the dimensionality of the feature set limited, so to make it possible to design the intermediate and combination classifiers as SVMs, without resorting to more complicated architectures such as ensemble classifiers [176]. Specifically, we selected the SPAM feature set, for which the number of features is reduced by exploiting symmetries, yielding a final dimension of the feature vector equal to 686. We refer to 2.4.1.2 for the details of the feature computation procedure. SPAM features are designed for grayscale images; when working with color images, they can be extracted from the luminance channel.

### 6.2.3 Training the 1.5C classifier

In this section, we describe the strategy we have adopted to train the intermediate and the combination classifiers. In general, training the 1.5C classifier is not easy, especially when the detection task is not straightforward, as it is often the case in image forensic applications. The difficulties are mainly associated to the one-class classifiers which are difficult to train and may not achieve good classification performance in many cases. The four classifiers of the 1.5C system are all SVMs whose hyper-parameters are determined by means of a preliminary internal validation phase. Let $(\boldsymbol{x}_i, y_i)$ be the training pair for image $I_i$, where $\boldsymbol{x}_i \in \mathbb{R}^r$ denotes the feature vector of dimensionality $r$ and $y_i$ denotes the class label associated to the image.

For a general intermediate SVM (hereafter indicated by the index $j \in \{2C_{H_{0/1}}, 1C_{H_0}, 1C_{H_1}\}$), the decision function $d_j(\boldsymbol{x})$ learned by the classifier is expressed as:

$$d_j(\boldsymbol{x}) = \sum_{i=1}^{n} y_i \alpha_{j,i} K_j(\boldsymbol{x}_i, \boldsymbol{x}) + b_j, \tag{6.1}$$

where $n$ is the number of training images, $K_j(\boldsymbol{x}_i, \boldsymbol{x})$ is the kernel function of the SVM, $b_j$ is the bias term, and $\alpha_j$ is a vector of scalars with $0 \leq \alpha_{j,i} \leq C_j$ where $C_j$ is the cost term, that is, the penalty parameter of the error term (over the training set) of the SVM optimization problem [42]. In our case, $\boldsymbol{x}_i$ is the SPAM feature vector of the $i$-th image; then, $r = 686$ and $\boldsymbol{x}_i \in \mathbb{R}^{686}$. For the $2C_{H_{0/1}}$ SVM, we set $y_i = 1$ for the images of the manipulated class and $y_i = 0$ for the pristine ones. For the 1C SVMs ($1C_{H_0}$, $1C_{H_1}$,and $1C_{H_0}^{cmb}$) instead, training is unlabeled, that is, the SVMs are trained with $(\boldsymbol{x}_i)$

only, and the decision function is given by (6.1) with $y_i = 1$, $\forall i$ [177]. In particular, we adopt an RBF (Radial Basis Function) kernel for the SVMs, that is, $K_j(\boldsymbol{x}_i, \boldsymbol{x}) = \exp(-\gamma_j \|\boldsymbol{x}_i - \boldsymbol{x}\|^2)$, where $\gamma_j$ determines the width of the Gaussian kernel.

In a similar way, the decision function of the final combination classifier ($1C_{H_0}^{cmb}$) is expressed by,

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_{f,i} K_f(\mathbf{d}(\boldsymbol{x}_i), \mathbf{d}(\boldsymbol{x})) + b_f, \tag{6.2}$$

where $\boldsymbol{f}d(\boldsymbol{x}) = (d_1(\boldsymbol{x}), d_2(\boldsymbol{x}), d_3(\boldsymbol{x}))$ is the vector of the soft outputs of the intermediate classifiers when the input feature vector is $\boldsymbol{x}$, $0 \leq \alpha_{f,i} \leq C_f$ and $C_f$ is the cost term. The kernel $K_f$ is again an RBF with parameter $\gamma_f$. The best parameters $\gamma_j^*$ and $C_j^*$ (and $\gamma_f^*$ and $C_f^*$) of the classifiers, often referred to as hyper-parameters or internal parameters, are determined during the validation phase.

Some observations are in order. In a 2C SVM, the parameter $C$ rules the tradeoff between the margin of the separating hyperplane in the higher dimensional space (the transformation of the input $\boldsymbol{x}$ to the higher-dimensional space defines the kernel [42]) and the misclassification of the training points. A large $C$ means that getting all the training points classified correctly, and then a smaller margin, is preferable, even if this goes with the risk of data overfitting. For 1C SVMs, according to the formulation by Schölkopf et al. [178], the selection of the hyper-parameters is conventionally carried out by considering $\gamma$ and $\nu = 1/C$ (rather than $\gamma$ and $C$), where $\nu$ determines the margin of the decision region in the higher-dimensional space. More specifically, the parameter $\nu$ sets an upper bound on the fraction of errors, i.e., training samples being misclassified [177] (for instance, by setting $\nu = 0.05$, at most 5% of the training samples are allowed to be wrongly classified).

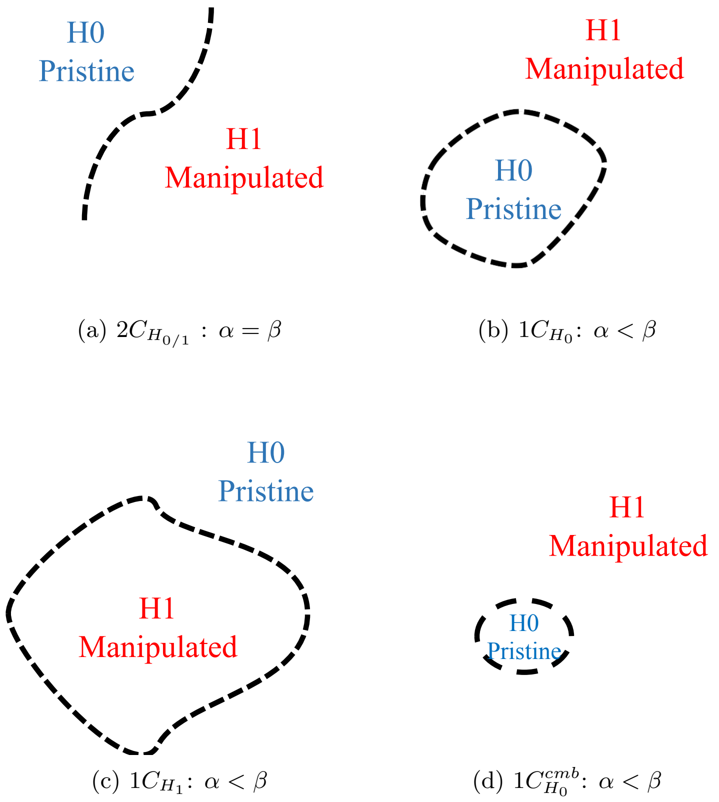
The most important parameter for both 2C and 1C SVMs in the case of RBF kernel is $\gamma$, which determines the width of the kernel and then determines how far the influence of a training sample reaches. Specifically, $\gamma$ defines the inverse of the radius of influence: the smaller is $\gamma$, the fewer support vectors are selected and the decision region becomes more spherical. Essentially, $\gamma$ regulates the tradeoff between capturing the complex shape of the data (large

$\gamma$) and avoiding overfitting (small $\gamma$). If $\gamma$ is too large, the radius of the area of influence of the support vectors includes *almost only* the support vector itself and no choice of the regularization terms $C$ and $\nu$ is able to prevent overfitting.

#### 6.2.3.1 Choice of the hyper-parameters of intermediate classifiers

We optimized the hyper-parameters of $2C_{H_{0/1}}$ on the validation set by means of an exhaustive search. To do so, we first split the validation set into a training and test set. Then we trained the system for every choice of the parameters $(C, \gamma)$, then we chose the pair with the best test accuracy, that is the pair that minimizes $P_e = 0.5P_{fa} + 0.5P_{md}$. For better performance in terms of generalization capability, standard $v$-fold cross validation was also performed for every $C$ and $\gamma$ [179], that is, we repeated the process $v$ times each time by splitting the set in a different way. The pair $(C, \gamma)$ with the best average cross-validation accuracy was then selected and used for training the system on the training set.

Similarly, for $1C_{H_0}$ and $1C_{H_1}$, we carried out an exhaustive search over both $\gamma$ and $\nu$ to find the pair leading to the best accuracy. However, the 1C classifiers tend by construction to have poor performance with respect to the alternative class, i.e., the class of samples not used for training. Since we wish to avoid missed detection events ($H_1$ detected as $H_0$), in order to increase the security against integrity violation attacks, we validated the 1C SVMs by weighting differently the two kinds of error probabilities. Let $\alpha$ and $\beta$ be the weights assigned to the probability of a false alarm and a missed detection, respectively. While for $2C_{H_{0/1}}$ we let $\alpha = \beta$, for the 1C classifiers we set $\alpha < \beta$ so that the classifiers are trained in such a way to minimize the error probability term $\alpha P_{fa} + \beta P_{md}$. This corresponds to consider a *relatively small* closed acceptance region for $1C_{H_0}$ and $1C_{H_0}^{cmb}$, and a *relatively large* closed region for $1C_{H_1}$. The situation is illustrated in Figure 6.4. According to the adversarial setup we are considered (see Figure 6.1(b)), in fact, the attacker aims at entering the pristine region, or equivalently, exiting the manipulated region; then, choosing $\alpha$ lower than $\beta$ should improve the performance of the 1C classifiers in the presence of attacks. Obviously, a considerable unbalance between the two weights may imply worse performance in the absence of



(a) $2C_{H_{0/1}}$ : $\alpha = \beta$

(b) $1C_{H_0}$: $\alpha < \beta$

(c) $1C_{H_1}$: $\alpha < \beta$

(d) $1C^{cmb}_{H_0}$: $\alpha < \beta$

Figure 6.4: Pictorial representation of the decision regions of the classifiers composing the overall 1.5C detector. To get an advantage in terms of security, the 1C classifiers are designed by letting $\alpha < \beta$. As a consequence, the 1C classifiers trained on pristine images - (b) and (d)- will have smaller acceptance regions, whereas the 1C trained on manipulated images - (c) - will have quite a large acceptance region.

attacks. However, we verified that, thanks to the presence of $2C_{H_{0/1}}$, the overall robustness of the 1.5C classifier remains good even when $\alpha$ is much lower than $\beta$ and then the 1C SVMs are designed by focusing more on the security performance, at the possible cost of a lower robustness.

### 6.2.4 Evaluation methodology

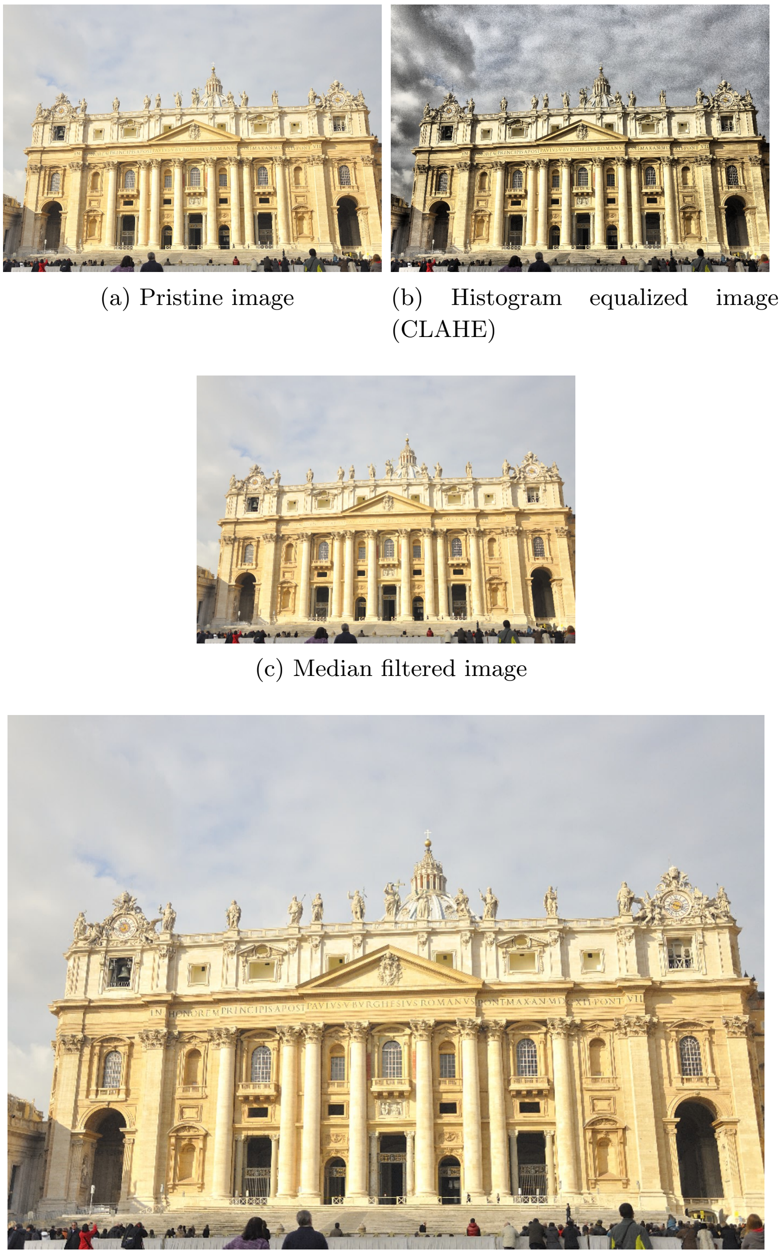
The goal of the 1.5C detector is to combine the good performance of a 2C classifier in the absence of intentional attacks (robustness) and the improved security achieved by 1C classification (security). To prove that this is indeed the case, we will show that the 1.5C detector is more robust than the intermediate 1C classifiers and that attacking the 1.5C detector requires a larger distortion with respect to attacking the intermediate 2C classifier. In the next subsections, we describe the exact methodology we have followed for our tests, while the results we have got are presented and discussed in Section 6.3.

### 6.2.5 Goal of the detectors

To asses the effectiveness of the 1.5C architecture, we focused on the detection of three different kinds of image manipulations, namely geometric transformation, filtering and contrast enhancement. Specifically, we considered the following three processing operations: resizing, median filtering and histogram equalization. With regard to resizing, we considered a bicubic interpolation and a resizing scaling factor (zooming factor) equal to 1.3.

For median filtering, we set the window size to $3 \times 3$, so to keep the visual degradation of the filtered image limited. Finally, for histogram equalization, we considered the Clip-Limited Adaptive Histogram Equalization (CLAHE) algorithm [155]. With respect to standard Adaptive Histogram Equalization (AHE), CLAHE does not over-amplify noise in relatively homogeneous regions as done by AHE, by clipping the histogram before computing the enhancement transformation. In our experiments, the clip-limit parameter was set to 0.05.

When working with color images, the CLAHE operator is applied to the luminance channel, precisely to the $V_{channel}$ (then, the image is converted from the RGB to the HSV color space). This is a commonly adopted strategy, since the straightforward application of CLAHE to each color channel separately would unnaturally change the color balance and produce a visually unpleasant image. The same strategy is followed for the case of median filtering, which, once again, is applied to the luminance channel only (the $V_{channel}$ in our case). An example of a pristine image and the corresponding processed images is given in Figure 6.5.



(a) Pristine image

(b) Histogram equalized image (CLAHE)

(c) Median filtered image

(d) Resized image

Figure 6.5: Visual comparison between a pristine image ($H_0$) and its manipulated versions ($H_1$), for the processing operators considered in this work. The clip-limit parameter for the CLAHE is 0.05; the windows size for the median filter is $3 \times 3$; resizing is applied with a zooming factor equal to 1.3.

### 6.2.6 Datasets creation

To produce the datasets for our experiments, we considered camera-native (uncompressed) images. The images of the dataset were used to train, validate and test the three intermediate classifiers ($2C_{H_{0/1}}$, $1C_{H_0}$,and $1C_{H_1}$). The set of images used for testing is further split to build the training and test sets for the final combination classifier $1C_{H_0}^{cmb}$. More specifically, let us denote with $S_V$ the set of images used for the internal validation of the hyper-parameters of the intermediate classifiers, where $S_{Tr}$ is the set used for training, and $S_T$ the set used for testing, see Figure 6.6.

The test set $S_T$ is further split into three sets, namely $S_T^v$, $S_T^{tr}$ and $S_T^t$, used, respectively, for internal validation, training and testing of $1C_{H_0}^{cmb}$. Since the dimensionality of the input feature vector of the downstream 1C is very low, corresponding to the three soft outputs of the intermediate classifiers ($\mathbf{d} \in \mathbb{R}^3$), the number of images in $S_T^v$, $S_T^{tr}$ and $S_T^t$ (and hence in $S_T$) does not need to be very large.

Starting from the above sets, no processing is applied to build the samples of the first class ($H_0$ - pristine images), see Figure 6.1(a). For the second class ($H_1$), the samples are built by applying different processing operators, as detailed in Section 6.2.5.

### 6.2.7 Robustness analysis

The robustness of the 1.5C detector is assessed by adopting the setup illustrated in Figure 6.1(a). Therefore, the system is trained and tested under the same conditions, i.e., by assuming that there is no attack at test time.

The performance of the 1.5C system are measured over the test set $S_T^t$, i.e., the set used for testing the final 1C classifiers. The metric used is the Area Under Curve (AUC) of the ROC curve of the classifier. By comparing these results with those achieved by $2C_{H_{0/1}}$ (which is tested on the entire $S_T$ set - see Figure 6.6), we can compare the performance of the 1.5C architecture with respect to a conventional 2C classifier, and assess the - possible - drop of performance experienced in the former case.[1] In general, 1C classifiers

[1] We verified experimentally that the fact that the performance of 2C and 1.5C are not tested exactly on the same set does not affect the results.

are known to have poor robustness in presence of post-processing, limiting their applicability in practice. To show that this is not the case with the overall 1.5C architecture, we also run some experiments to assess the robustness performance of the 1.5C classifier in the presence of noise addition and JPEG post-processing. Results confirm that, thanks to the presence of the intermediate 2C classifier, the 1.5C system performs much better than the intermediate 1C detectors in terms of robustness. For sake of simplicity, for these tests, we only considered the case of resizing detection. The exact results of our experiments will be detailed in Section 6.3.

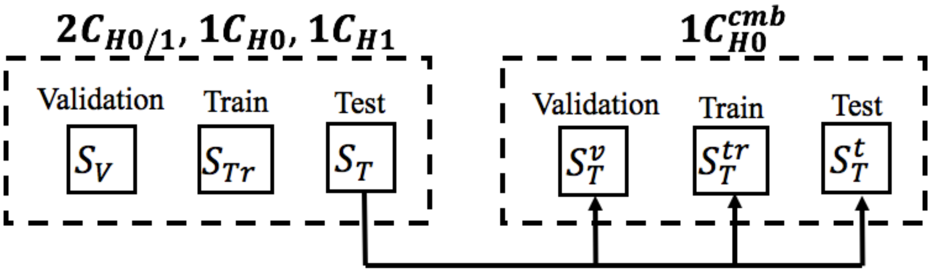


Figure 6.6: Datasets used for training and testing the classifiers of the 1.5C system. $\{S_T^v \cup S_T^{tr} \cup S_T^t\} \equiv S_T$.

### 6.2.8 Security assessment

The security of the 1.5C classifier is assessed by evaluating the performance of the system under attacks (see Figure 6.1(b)). These performance are compared against those achieved by $2C_{H_{0/1}}$, when tested under the same attack. The goal of this analysis is to validate the expectation that the 1.5C architecture offers a security advantage over 2C classification, in that attacking the system introduces a larger distortion into the attacked images.

In our scenario, the attacker aims at inducing a missed detection error, by minimizing the distortion introduced into the image. It is worth stressing that, a targeted PK attack is always successful in causing a misclassification, i.e., it always enter the $H_0$ region. However, it is expected that attacking a more secure classifier will require a larger distortion.

Regarding the attack algorithm, we considered the gradient-based attack against SVM detectors described in [7]. The attack works by computing an approximation of the gradient of the SVM output with respect to the image pixels. The approximated gradient provides an approximation of the steepest descent direction of the decision function. Then, the step size (strength of the attack) is adjusted by controlling the percentage of modified pixels. If the image cannot be successfully attacked by modifying a maximum prescribed fraction of pixels, the modification is applied and the process is iterated. As opposed to other approaches, the attack proposed in [7] is directly carried out in the pixel domain and then it can be applied even when the relationship between the feature and the pixel domain is not invertible, as it is the case with the SPAM features. The implementation of the attack passes through the definition of a safety margin $\rho$ [7], which determines how much the attack goes inside the acceptance region. By choosing a larger $\rho$ (so to move the attacked image more deeply inside the $H_0$ region), the attack is more robust to perturbations of the decision boundary. This advantage goes at the price of a larger distortion introduced in the image.

In order to compare the security of the 1.5C classifier with respect to the 2C one, we run the targeted attack in [7] against $2C_{H_{0/1}}$ and $1C_{H_0}^{cmb}$. The performance of the classifiers under attacks are assessed on a subset of images in $S_T^t$, processed with median filtering, resizing and CLAHE and then attacked by means of the attack described above.

## 6.3 Experimental Results

The camera-native (uncompressed) images used for our experiments were taken from the RAISE-8K dataset. Specifically, a total amount of 7997 images were used, split as follows (see Figure 6.6): 1000 images were selected to build the validation set $\mathcal{S}_V$, 5000 for the training set $\mathcal{S}_{Tr}$, and the remaining 1997 for $\mathcal{S}_T$. Then, the images in $\mathcal{S}_T$ were further split to build the validation, training and test sets used for the combination classifier as follows: 300 images were used to build $\mathcal{S}_T^v$, 700 for $\mathcal{S}_T^{tr}$ and the remaining 997 images to build $\mathcal{S}_T^t$. Note that a much lower number of images would be sufficient for testing the final SVM which is trained on just 3-dimensional input feature vectors. The images from every set were then processed to build the class of $H_1$ samples, whereas the unprocessed images were used to build the class of pristine images ($H_0$). For security assessment, the attack in [7] was applied to 100 images in the set $\mathcal{S}_T^t$ belonging to the $H_1$ class. To speed up the feature extraction step and the attacks, we sub-sampled the images from the RAISE-8K dataset down to a size of about $1072 \times 770$ without interpolation.

The Matlab environment was used to process the images and to design the classifiers of the 1.5C system. All SVMs were trained and tested by using the LibSVM library package [162]. We run our experiments on a system hardware Intel(R) Core i7-6700 CPU @ 3.40 GHz with four cores, and 32 GB of RAM.

### 6.3.1 Hyper-parameters setting

As anticipated in Section 6.2.2, the SPAM features are extracted from the V channel, obtained by converting the image from the RGB to the HSV color space. Regarding the weights assigned to the two types of error probabilities during the validations phase, we set $\alpha = 0.2$ ($\beta = 0.8$) for $1C_{H_0}$ and $1C_{H_1}$ and $\alpha = 0.1$ ($\beta = 0.9$) for $1C_{H_0}^{cmb}$. In making this choice of $\alpha$ and $\beta$, we verified that the system robustness is not affected too much by the use of heavily asymmetric weights. For validating the internal parameters of $2C_{H_{0/1}}$, we followed the standard exhaustive search method (known as grid-search) in the LibSVM library [162], which considers exponentially growing values for $C$ and $\gamma$ to identify the best parameters [180]. Specifically, we considered the following grid-search area: $C \in \{2^{-5}, 2^{-3}, ..., 2^{15}\}$ and $\gamma \in \{2^{-15}, 2^{-13}, ..., 2^{3}\}$ and performed a 5-fold cross validation (i.e. $\nu = 5$).

To set the hyper-parameters of $1C_{H_0}$, $1C_{H_1}$, and $1C_{H_0}^{cmb}$, we followed a similar strategy by taking $\nu$, $\gamma \in \{2^{-10}, 2^{-9}, ..., 2^{9}, 2^{10}\}$. In addition, since the distribution of the samples used for internal parameter validation is very important to learn correctly the hyper-parameters of the 1C SVMs, we used the entire training set to train the SVMs during the exhaustive search; then, the validation set was used only to perform testing in this phase and choose the pair $(\gamma, \nu)$ providing the best accuracy. To limit the computational burden, $v$-fold cross validation was not performed in this case.

Table 6.1 shows the best hyper-parameters ($C^*$, $\gamma^*$) for $2C_{H_{0/1}}$ and ($\nu^*$,$\gamma^*$) for the three 1C SVMs. From the tables, we see that, for the $1C_{H_0}^{cmb}$ SVM,

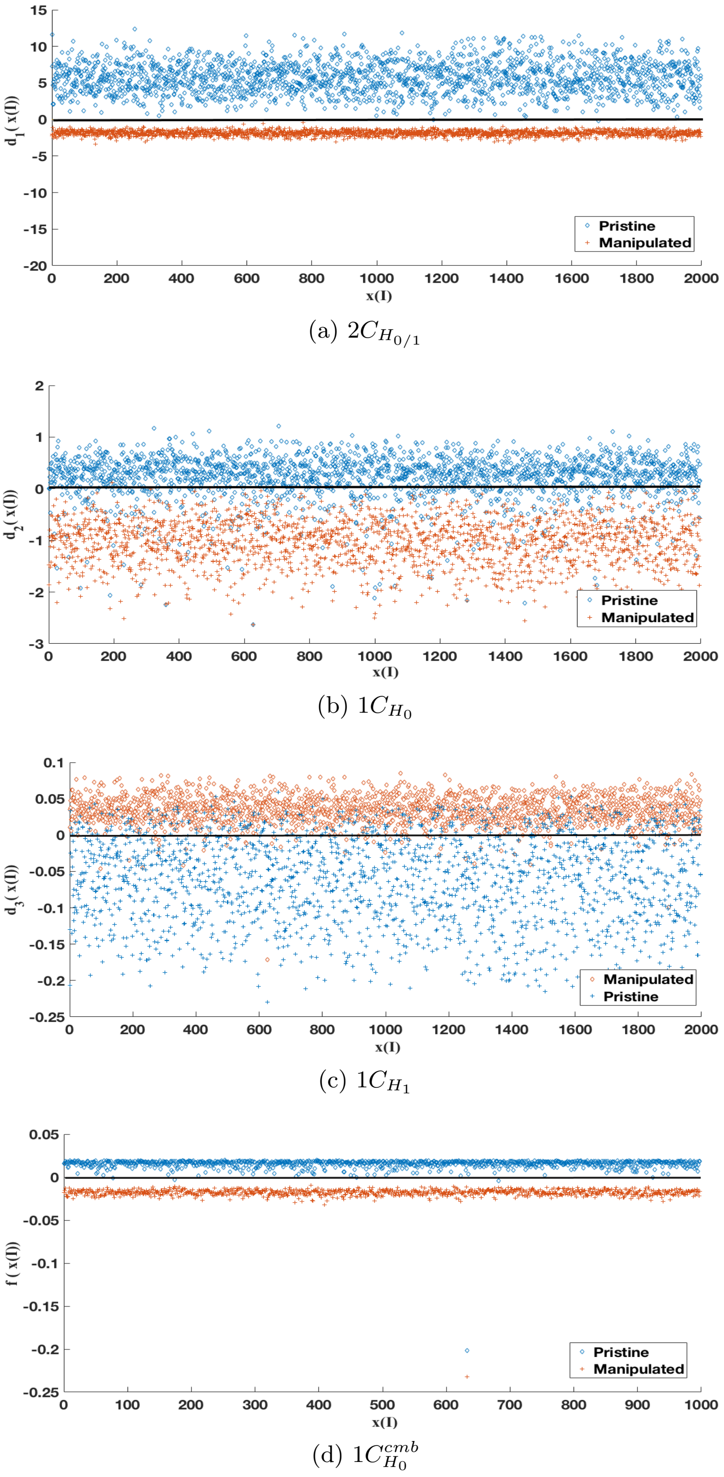
Table 6.1: Hyper-parameters of the SVMs classifiers.

| | $2C_{H_{0/1}}$ | $1C_{H_0}$ | $1C_{H_1}$ | $1C_{H_0}^{cmb}$ |
|---|---|---|---|---|
| Resizing | $C^* = 2^{11}$<br>$\gamma^* = 2^{-1}$ | $\nu^* = 2^{-3}$<br>$\gamma^* = 2^{9}$ | $\nu^* = 2^{-9}$<br>$\gamma^* = 2^{7}$ | $\nu^* = 2^{-10}$<br>$\gamma^* = 2^{-10}$ |
| Median filter | $C^* = 2^{11}$<br>$\gamma^* = 2^{-1}$ | $\nu^* = 2^{-5}$<br>$\gamma^* = 2^{7}$ | $\nu^* = 2^{-9}$<br>$\gamma^* = 2^{6}$ | $\nu^* = 2^{-10}$<br>$\gamma^* = 2^{-10}$ |
| CLAHE | $C^* = 2^{11}$<br>$\gamma^* = 2^{-1}$ | $\nu^* = 2^{-3}$<br>$\gamma^* = 2^{9}$ | $\nu^* = 2^{-10}$<br>$\gamma^* = 2^{7}$ | $\nu^* = 2^{-10}$<br>$\gamma^* = 2^{-10}$ |

the minimum values of $\nu$ and $\gamma$ are selected, meaning that the SVM is able to get a low probability of erroneous classification of the training samples by relying on very few support vectors, hence minimizing the risk of overfitting.

### 6.3.2 Performance in the absence of attacks

The values of the decision functions of the four SVM classifiers over the test set are reported in Figure 6.7 for the case of resizing detection. We see that $2C_{H_{0/1}}$ is able to tell apart pristine and manipulated images, obtaining perfect classification in the absence of attacks; moreover, the scatter plot shows that the clouds of points are very well separated. The two intermediate 1C SVMs also achieve high-accuracy, but the classification is not perfect. Finally, $1C_{H_0}^{cmb}$ achieves almost perfect classification, similarly to $2C_{H_{0/1}}$. For both $2C_{H_{0/1}}$ and $1C_{H_0}^{cmb}$, the decision threshold is set to 0. Note that while for $2C_{H_{0/1}}$, $1C_{H_0}$ and $1C_{H_0}^{cmb}$, the label $y = 1$ is assigned to the images of the pristine class ($H_0$), for $1C_{H_1}$, $y = 1$ is assigned to the manipulated class ($H_1$), and that is why the scatter plots in Figure 6.7c are reverted. Very similar results were obtained for median filtering and CLAHE. Table 6.2 shows the AUC values of the ROC curve for $2C_{H_{0/1}}$ and the 1.5C classifiers, as well as those of the intermediate 1C SVMs, for each detection task. We observe that by using $1C_{H_0}^{cmb}$, instead of $2C_{H_{0/1}}$, the performance drops very slightly in all the cases.



(a) $2C_{H_{0/1}}$

(b) $1C_{H_0}$

(c) $1C_{H_1}$

(d) $1C_{H_0}^{cmb}$

Figure 6.7: Decision values of the four SVMs trained for resizing detection on the test set, for both $H_0$ (pristine samples) and $H_1$ (manipulated samples). Decision values of $2C_{H_{0/1}}$ for the images in $S_T$ (a); decision values of the intermediate 1C classifiers ($1C_{H_0}$,and $1C_{H_1}$) for the images in $S_T$ (b)-(c); decision values of $1C_{H_0}^{cmb}$ for the images in $S_T^t$ (d).

Table 6.2: AUC values of all the classifiers for the three manipulation detection tasks. The performance of the 1.5C system are those reported for $1C_{H_0}^{cmb}$.

| | $2C_{H_{0/1}}$ | $1C_{H_0}$ | $1C_{H_1}$ | $1C_{H_0}^{cmb}$ |
|---|---|---|---|---|
| Resizing | **1** | 0.96 | 0.96 | **0.99** |
| Median filter | **1** | 0.99 | 0.99 | **0.99** |
| CLAHE | **1** | 0.95 | 0.97 | **0.99** |

#### 6.3.2.1 Robustness of the classifiers

To assess the robustness of the 1.5C architecture compared to the 2C and the 1C detectors, we evaluated the performance of $2C_{H_{0/1}}$, $1C_{H_0}$, $1C_{H_1}$ and $1C_{H_0}^{cmb}$ in the presence of Gaussian noise with zero mean and variance $\sigma^2 = 5 \cdot 10^{-6}$, $10^{-5}$, $1.5 \cdot 10^{-5}$ and $2 \cdot 10^{-5}$ (standard deviation ranging from $\sigma = 0.0022$ to $\sigma = 0.0045$), and in the presence of JPEG compression with Quality Factors (QF) 85, 90, 95, and 98. For the case of noise addition, the average Mean Square Error (MSE) introduced by the noise ranges from 0.3 to 1.

Tables 6.3 and 6.4 show the average accuracy of the tests on noisy images and JPEG compressed images respectively, for the resizing detection task. We see that while the performance of the 1Cs classifiers are significantly impaired by the post-processing, the 1.5C classifier is more robust and its performance remain comparable to those of the 2C detector.

Table 6.3: Robustness of the classifiers in the presence of JPEG compression (accuracy).

| QF | $2C_{H_{0/1}}$ | $1C_{H_0}$ | $1C_{H_1}$ | $1C_{H_0}^{cmb}$ |
|---|---|---|---|---|
| 85 | 0.90 | 0.71 | 0.83 | 0.88 |
| 90 | 0.94 | 0.75 | 0.87 | 0.93 |
| 95 | 0.98 | 0.82 | 0.90 | 0.97 |
| 98 | 0.99 | 0.87 | 0.91 | 0.99 |

Table 6.4: Robustness of the classifiers under noise addition (accuracy).

| Noise Parameter | $2C_{H_{0/1}}$ | $1C_{H_0}$ | $1C_{H_1}$ | $1C^{cmb}_{H_0}$ |
|---|---|---|---|---|
| $5 \cdot 10^{-6}$ | 0.87 | 0.78 | 0.84 | 0.93 |
| $10^{-5}$ | 0.84 | 0.79 | 0.85 | 0.92 |
| $1.5 \cdot 10^{-5}$ | 0.79 | 0.80 | 0.87 | 0.89 |
| $2 \cdot 10^{-5}$ | 0.74 | 0.82 | 0.89 | 0.83 |

Expectedly, if we consider a larger noise (or a stronger compression), the performance of the classifiers drop. In order to design a classifier that works properly under these conditions, a possibility is to consider an aware classifier, which takes into account the possible presence of post-processing during the training phase, by including post-processed samples in the training set [133]. This analysis is outside the scope of this work, since here we are interested in validating the 1.5C architecture, so we leave it for a future work.

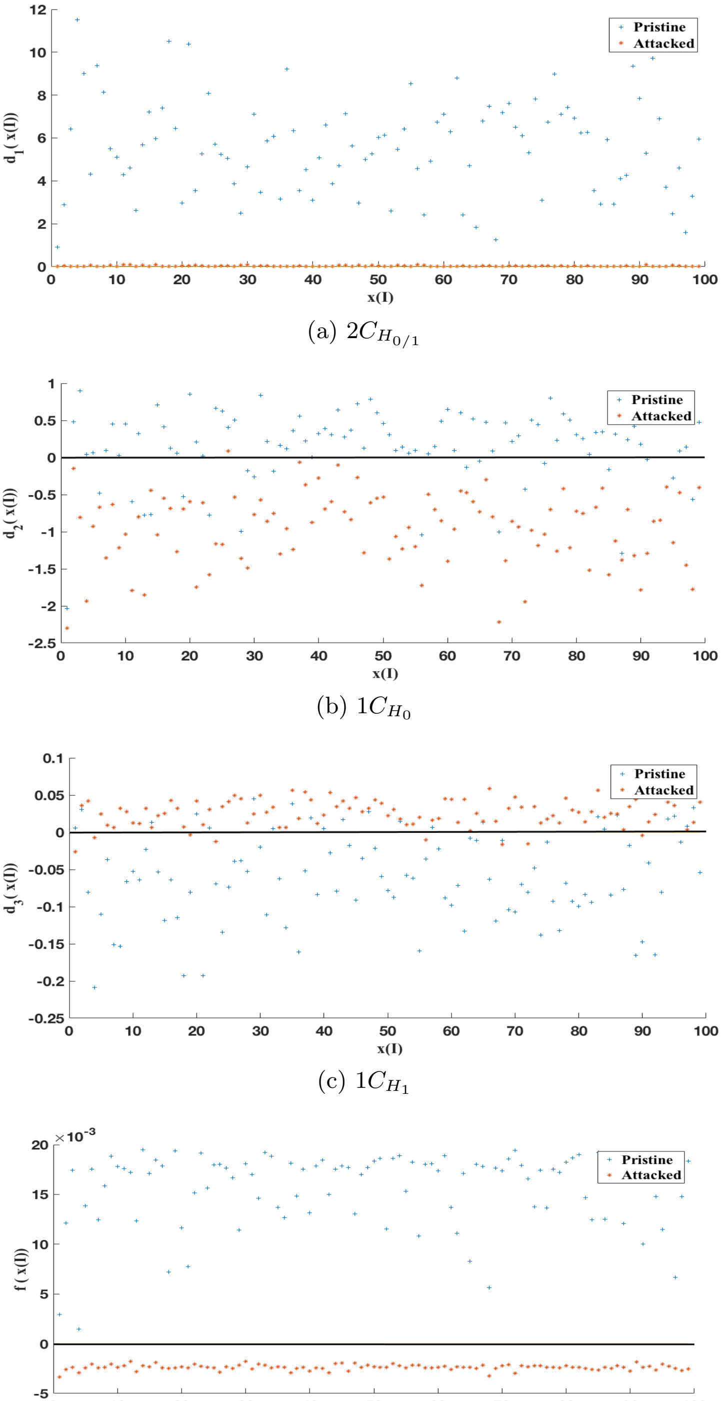
### 6.3.3 Performance under attacks

In this section, we assess the performance of $2C_{H_{0/1}}$ and the 1.5C classifiers in the presence of attacks [7]. In all the experiments, the safety margin $\rho$ for the attack is set to 0.

#### 6.3.3.1 Attack against $2C_{H_{0/1}}$

For each detection task, we first run the attack against $2C_{H_{0/1}}$. As expected, the attack is always successful in inducing an incorrect classification and 100% of the manipulated images are classified as pristine images after the attack. Moreover, all the images can be attacked in just one iteration of the algorithm in [7]. Figure 6.8a shows the results for the case of resizing detection. Since we set $\rho = 0$, the attack stops as soon as the decision boundary is crossed. The values of the decision function for the other SVMs of the 1.5C c classifier are shown in Figure 6.8 (from 6.8b to 6.8d)[2]. From Figure 6.8d, we see that

[2] Note that, while for $2C_{H_{0/1}}$, $1C_{H_0}$ and $1C^{cmb}_{H_0}$ the attack is successful when it brings the pristine samples above the threshold, for $1C_{H_1}$, the goal of the attacker is to move the



(a) $2C_{H_{0/1}}$

(b) $1C_{H_0}$

(c) $1C_{H_1}$

(d) $1C^{cmb}_{H_0}$

Figure 6.8: Decision values of the four SVMs on the 100 images in $S^t_T$ in the presence of the attack in [7] under $H_1$, for the resizing detection task. The attack is carried out against the $2C_{H_{0/1}}$. Decision values of $2C_{H_{0/1}}$ (a); decision values of the intermediate 1C classifiers ($1C_{H_0}$, and $1C_{H_1}$) (b)-(c); decision values of $1C^{cmb}_{H_0}$ (d).

attacking $2C_{H_{0/1}}$ is not enough to fool the 1.5C classifier: the attacked samples in fact remain much below the decision threshold and the attack success rate is 0%. The success rate of all the attacks is reported in Table 6.5, where the percentage of misclassified attacked samples for the four SVMs is provided for the three detection tasks. Figure 6.9 shows an example of processed image for the three different tasks, before and after the attack against the $2C_{H_{0/1}}$ classifier.

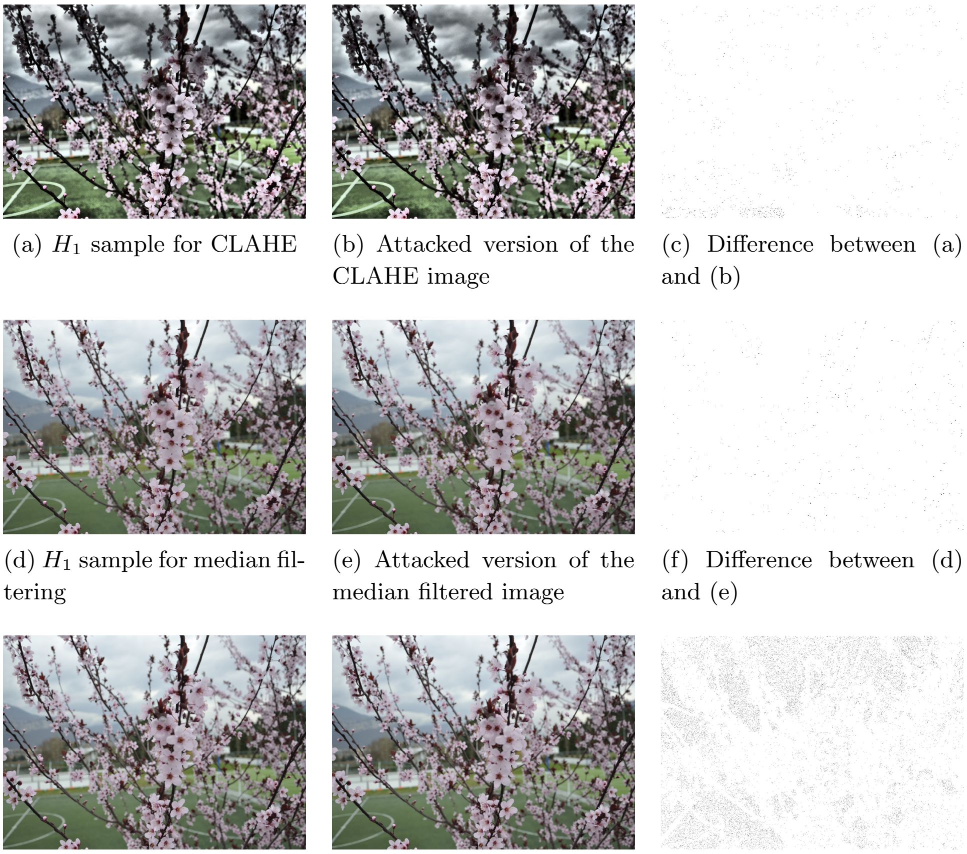
Table 6.5: Percentage of misclassified attacked images. The attack is carried out against $2C_{H_{0/1}}$.

| | $2C_{H_{0/1}}$ | $1C_{H_0}$ | $1C_{H_1}$ | $1C^{cmb}_{H_0}$ |
|---|---|---|---|---|
| Resizing | **100%** | 1% | 8% | **0%** |
| Median Filter | **100%** | 4% | 3% | **0%** |
| CLAHE | **100%** | 20% | 12% | **0%** |

#### 6.3.3.2 Attack against the 1.5C classifier

Figures 6.10 shows what happens when the attack is carried out against the 1.5C classifier. In this case, most of the times, the attack requires more than one iteration to enter the $H_0$ region. The figure refers to the case of resizing detection, however, similar results are obtained for the other manipulations. We observe that the values of the decision function for $2C_{H_{0/1}}$ on the attacked samples lie above the 0 threshold, and then the attack against the 1.5C is also effective against $2C_{H_{0/1}}$. Moreover, we see that the attack is not much effective against $1C_{H_0}$, and quite ineffective against $1C_{H_1}$, thus confirming that, thanks to the adoption of a closed acceptance region, the 1C classifiers are more difficult to attack. Accordingly, the attack is successful in inducing a wrong classification for the 1.5C, mainly because $2C_{H_{0/1}}$ fails to a strong extent. This suggests that, in order to be successful against the 1.5C detector, the attack has to introduce a larger distortion into the image. The attack success rate against the four SVMs is reported in Table 6.6, for all the detection tasks.

samples below the threshold.



(a) $H_1$ sample for CLAHE

(b) Attacked version of the CLAHE image

(c) Difference between (a) and (b)

(d) $H_1$ sample for median filtering

(e) Attacked version of the median filtered image

(f) Difference between (d) and (e)

(g) $H_1$ sample for resizing

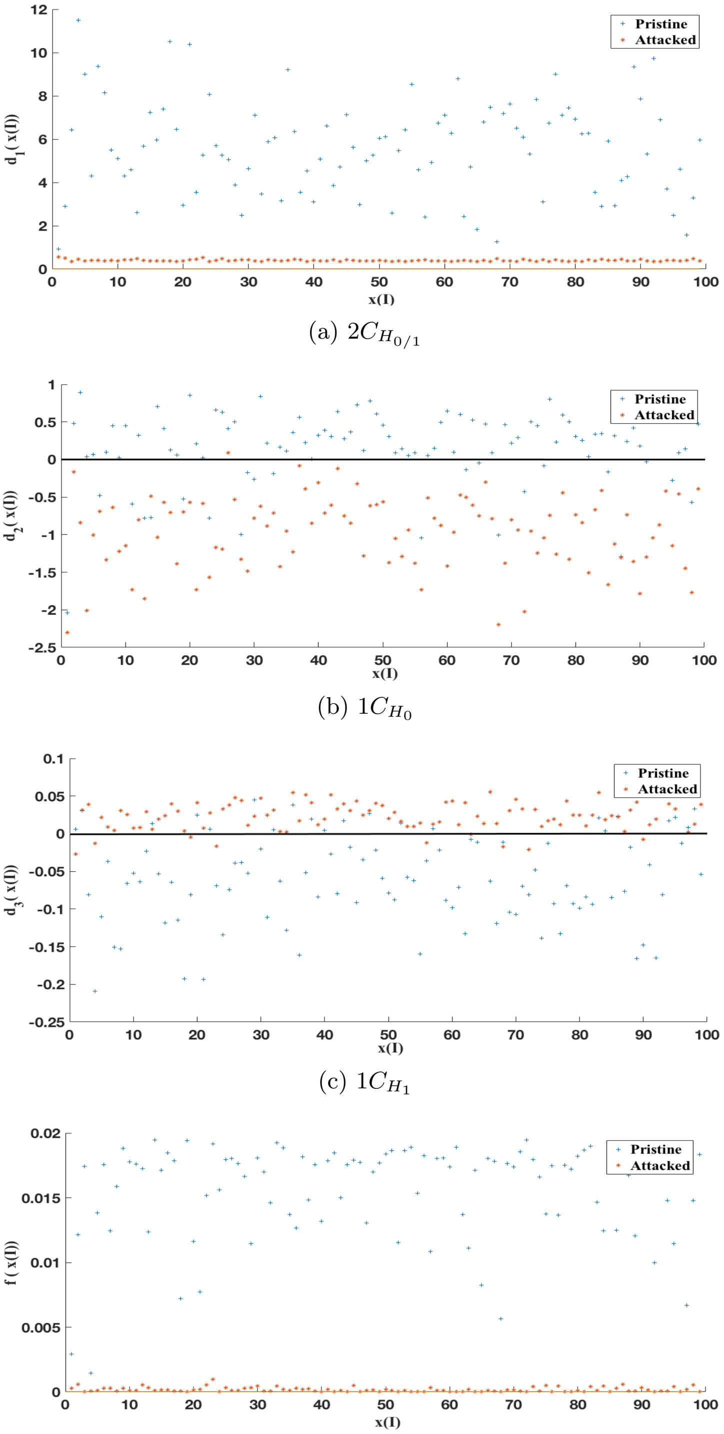
(h) Attacked version of the resized image

(i) Difference between (g) and (h)

Figure 6.9: Visual comparison of processed image before and after the attack to the $2C_{H_{0/1}}$.

Figure 6.11 shows an example of processed image for the three different tasks, before and after the attack against the $1C_{H_0}^{cmb}$ classifier.

Table 6.7 compares the attack against $2C_{H_{0/1}}$ and the 1.5C detector in terms of MSE of the attack. Specifically, the MSE averaged on the 100 attacked images in $\mathcal{S}_T^t$ is reported in the table for the two attacks. The values are referred to the image range $(0 : 255)$. We see that, in order to make the



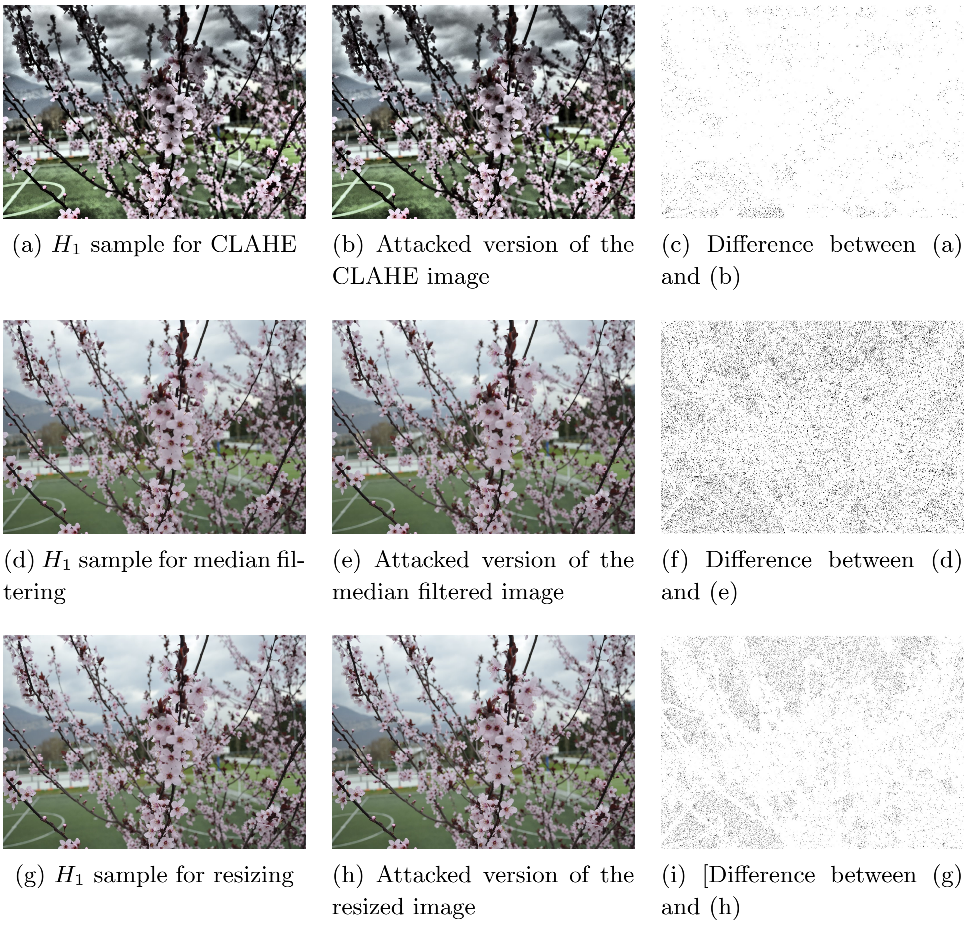
(a) $2C_{H_{0/1}}$

(b) $1C_{H_0}$

(c) $1C_{H_1}$

(d) $1C_{H_0}^{cmb}$

Figure 6.10: Decision values of the four SVMs on the 100 images in $S_T^t$ in the presence of the attack in [7] under $H_1$, for the resizing detection task. The attack is carried out against the 1.5C classifier. Decision values of $2C_{H_{0/1}}$ (a); decision values of the intermediate 1C classifiers ($1C_{H_0}$,and $1C_{H_1}$) (b)-(c); decision values of $1C_{H_0}^{cmb}$ (d).



(a) $H_1$ sample for CLAHE

(b) Attacked version of the CLAHE image

(c) Difference between (a) and (b)

(d) $H_1$ sample for median filtering

(e) Attacked version of the median filtered image

(f) Difference between (d) and (e)

(g) $H_1$ sample for resizing

(h) Attacked version of the resized image

(i) [Difference between (g) and (h)

Figure 6.11: Visual comparison of processed image before and after the attack to the $1C_{H_0}^{cmb}$.

1.5C classifier fail, the attacker must introduce a larger MSE with respect to the case in which the targeted classifier is $2C_{H_{0/1}}$: the average value of the MSE in the case of the attack against the 1.5C detector is more than twice that necessary for the case of resizing and contrast enhancement and almost double for the case of median filtering. The average percentage of pixels modified by the two attacks are reported in Table 6.8. The table confirms that, in

Table 6.6: Percentage of misclassified attacked images. The attack is carried out against $1C_{H_0}^{cmb}$.

| | $2C_{H_{0/1}}$ | $1C_{H_0}$ | $1C_{H_1}$ | $1C_{H_0}^{cmb}$ |
|---|---|---|---|---|
| Resizing | **100%** | 1% | 9% | **100%** |
| Median Filter | **100%** | 20% | 21% | **100%** |
| CLAHE | **100%** | 23% | 14% | **100%** |

Table 6.7: Average MSE.

| | Resizing | median Filter | CLAHE |
|---|---|---|---|
| Attack against $2C_{H_{0/1}}$ | 0.10 | 0.22 | 0.27 |
| Attack against $1C_{H_0}^{cmb}$ | 0.17 | 0.60 | 0.43 |

order to be successful against the 1.5C classifier, the attacker has to modify a larger number of pixels. The average Structural Similarity (SSIM) index [181] has also been reported in Table 6.9.

Table 6.8: Average percentage of pixels modified by the attack.

| | Resizing | median Filter | CLAHE |
|---|---|---|---|
| Attack against $2C_{H_{0/1}}$ | 9.5% | 15.1% | 12.3% |
| Attack against $1C_{H_0}^{cmb}$ | 13.4% | 25.1% | 15.1% |

### 6.3.4 Comparison with 2C classifier based on Convolutional Neural Networks (CNNs)

In order to assess the performance of the proposed system with respect to state of the art CNN-based image manipulation detection, in this section, we consider the case of CNN-based 2C classification in the presence of a targeted

Table 6.9: Average SSIM.

| | Resizing | median Filter | CLAHE |
|---|---|---|---|
| Attack against $2C_{H_{0/1}}$ | 0.9794 | 0.975 | 0.9920 |
| Attack against $1C_{H_0}^{cmb}$ | 0.9609 | 0.958 | 0.9868 |

attack. For these experiments, we considered the CNN architecture in [8], used in the literature for several manipulation detection tasks. We set the input patch size to 128×128. The datasets for training and testing the models were obtained by splitting into blocks the images in $S_{Tr}$ and $S_V$ for training (and validation) and $S_T$ for testing. The same parameters setting used in [8] has been adopted for training the models (optimization solver, learning rate, batch size, etc ...). All the models were trained on 20 epochs. The average test accuracy of the trained CNN models are 97.8%, 81% and 86.9% for resizing, median filtering and CLAHE respectively.[3] Given a test image, the decision is made by dividing the image into non-overlapping patches, testing each patch with the trained model, and then fusing the CNN outputs. For simplicity, the normalized sum of the decision scores ('0' for original, '1' for manipulated) is considered as the final score for the entire image. Then, for a given an image, the decision is made by thresholding the accumulated score. The performance of the classification are measured again by relying on the ROC curve obtained by varying the decision threshold: in particular we got AUC=1 for resizing, AUC =0.98 for median filtering and AUC =0.93 for CLAHE.

To attack the CNN classifiers, we considered the well known Jacobian-based Saliency Map Attack (JSMA) method [119], due to its good effectiveness even in the presence of integer rounding. To keep the distortion low, the attack is applied with the following setting: the relative amount of pixel modification ($\theta$) is set to 0.005; the maximum number of times the same pixel can be modified is set to 3; finally, the maximum number of iterations for the

[3] The detection median filtering with a small (3 × 3) window, as well as the detection of CLAHE, are not easy tasks. Deeper networks could give better performance in this cases; however, this goes at the price of lower robustness against attacks, as deeper models are known to be more vulnerable to attacks than shallow ones [117].

attack is set to 8000. We verified that similar performance are obtained by using the pixel-domain attack in [182] which extends the attack in [7] developed for the SVMs, to the case of CNNs. As before, the attack is applied to 100 of images from the test set $S_T^t$, belonging to the $H_1$ class. In order to be successful, the attack should be able to fool the CNN model, and then revert the decision, for at least half of the patches of the image. To minimize the overall distortion, the 'most favorable' patches are considered by the attack, that is, those patches that can be attacked by introducing the minimum (MSE) distortion.

The final average MSE of the attack, averaged on all the 100 images, was: 0.055 for resizing, 0.094 for median filtering and 0.347 for CLAHE. We observe that these MSE values are lower than those obtained with the 2C SVM for the case of resize and median filtering detection, and always lower (significantly lower) than those for the 1.5C case (see Table 6.7). This is not surprising, since it is known that CNNs are vulnerable to adversarial attacks and can be attacked by introducing very small perturbations. Assessing the security gain that can be obtained by using CNNs to build a 1.5C classifier is an interesting piece of work, and will be considered as a future research.

## 6.4 Concluding Remarks

In this chapter, we have described to use a multiple classifier architecture, referred to as 1.5C classifier, to mitigate the damage made by an attacker with perfect knowledge acting against an image manipulation detector. In such a situation, the only possible defence for the analyst is to use a detector which is intrinsically more difficult to attack. This is the case of 1C classifiers, which, however, have the drawback of achieving inferior performance with respect to more conventional 2C classifiers. By properly combining one 2C classifier and three 1C classifiers, the 1.5C classifier couples the advantages of 2C and 1C solutions, achieving a superior security while retaining the good performance of 2C classification in the absence of attacks. We implemented a particular instantiation of the proposed architecture by relying on four SVMs, and we trained it so to detect three kinds of image manipulations, namely median filtering, resizing and adaptive histogram equalization.

The experimental analysis we carried out confirms that the 1.5C architecture is harder to attack than a 2C classifier with similar performance.

Chapter 7

# On the Transferability of Adversarial Examples Against CNN-Based Image Forensics

*"Securing a computer system has traditionally been a battle of wits: the penetrator tries to find the holes, and the designer tries to close them."*
Gosser

Recent research in Deep Learning (DL) has shown that adversarial examples against Convolutional Neural Network (CNN) classifiers present a certain degree of transferability, i.e., they maintain part of their effectiveness even against CNN models other than the one considered for the attack. In this chapter, we investigate if and to which extent *transferability* holds against CNN models developed for Image Forensic applications.

The chapter is organised as follows: in Section 7.1, we discuss the reason behind the investigation we carried out, pointing out the consequences of the transferability attacks. A brief introduction to the most common adversarial attacks against DL classifiers is provided in Section 7.2. Then, in Section 7.3, we describe the methodology used for our experiments, including the algorithms used to generate the adversarial examples, the CNN architectures targeted by the attacks, the description of the experimental campaign, and the datasets used for training and testing the CNNs. The results of the experiments are presented in Section 7.4, together with a discussion of our main findings.

## 7.1 Motivation and Contribution

As we said in the introductory part of this thesis (Chapter 2) Convolutional Neural Networks (CNN) are increasingly used in a wide variety of image forensic applications due to their superior performance compared to standard machine learning techniques (e.g., SVMs, NNs,...). However, as we already mentioned in Chapter 3, such widespread use of CNNs is hindered by the easiness with which adversarial attacks, namely *adversarial examples*, can be built [117, 119, 183]. As a matter of fact, an attacker who has access to the internal details of the CNN used for a certain image recognition task can easily build an attacked image which is visually indistinguishable from the original one, but is misclassified by the CNN. Such a problem is currently the subject of an intense research activity, yet no satisfactory solution has been found yet (see [184] for a recent survey on this topic). The problem is worsened by the observation that adversarial attacks, carried out against a given network, are often transferable to other networks designed for the same task [13]. This means that even in a Limited Knowledge (LK) scenario, wherein the attacker has only partial information about the to-be-attacked network, he can attack a surrogate network mimicking the target one and the attack will be effective also on the target network with large probability. Such a property opens the way towards very powerful attacks that can be used in real applications, where the attacker does not have full access to the attacked system.

Given that CNN-based image forensic tools are also endangered by the existence of adversarial examples, as shown by many recent literature (see Chapter 3, Section 3.2.4), the issue of the transferability of such adversarial examples in image forensics applications is worth investigation. In fact, even denying to the attacker a full access to the forensic tools would not guarantee that the forger can not mislead the forensic analysis. Prior to our work, this problem was only partially addressed by some few scattered works, like, [116] and [123]. In particular, in [116] reports some tests aiming at assessing the transferability of adversarial examples targeting various CNN-based camera model identification systems. According to [116], in a camera model identification scenario, attacks are only partially transferable, since the transferred attack succeed in no more than 40% of the cases (often much less). In [123], the transferability between different network models is assessed by considering a specific adversarial attack, named FGSM [14], which is a fast, yet suboptimum, attack method (see Section 7.2 for an introduction to the attacks to CNN models). The analyses carried out in the above works are very preliminary, hence calling for extensive tests and investigation addressing different sources of mismatch between the attacked network and the targeted one, different forensics scenarios, and the impact that the attack strength has on the transferability of the attacks, which is the contribution of this chapter. Specifically, we consider two forensic tasks boiling down to a binary detection problem, namely, median filtering and image resizing detection.

As we will see, our experiments cast serious doubts on the transferability of adversarial attacks against CNN models in image forensic applications, thus opening the way to the development of proper countermeasures, enforcing a LK scenario (e.g., the feature randomization approach considered in the next chapter).

## 7.2 Attack against DL models

In general, the attacks tailored to deep learning CNN models can be split into two groups, targeted and untargeted attacks. Targeted attacks aim at fooling a deep-learning model by inducing the model to output a specific target label for the adversarial image, while untargeted attacks aim at letting the trained network to misclassify the input, regardless of the output label. Obviously, for a binary classification or detection task (which is the case we focus on in this thesis), there is no difference between targeted and untargeted attacks.

With reference to binary classification, in the following, we present some of the most famous and common adversarial attacks to CNN models proposed in the DL literature. These attacks are all gradient-based methods, designed for a white-box scenario (the CNN model used to craft the adversarial image is fully known to the attacker).

### 7.2.1 L-BFGS

Authors in [117] generated adversarial examples using box-constrained L-BFGS.

Given an image $X$, the method looks for an image $X_{adv}$, that is similar to $X$ under the $L_2$ distance, yet is labeled differently by the classifier. Let $y$ be the ground truth class of $X$, then, $X_{adv}$ is such that $l(X_{adv}) \neq y$, that is, $l(X_{adv}) = 1 - y$, where $l()$ denotes the class label (then, $l(X_{adv}) = 0$ if $y = 1$, and viceversa). The constrained minimization problem is formalized as

$$\begin{aligned} \min_{X'} \quad & ||X - X'||_2^2 \\ s.t. \quad & l(X') = 1 - y. \end{aligned} \tag{7.1}$$

This problem is very difficult to solve. An approximately optimum solution is then found in [117] by solving the following relaxed problem:

$$\min_{X'} \quad c||X - X'||_2^2 - J_\theta(X, y), \tag{7.2}$$

where $J_\theta(X, y)$ denotes the loss function (e.g., the cross-entropy) of the neural network model with parameters $\theta$, and $c$ is a scalar. The gradient-descent algorithm is applied to solve (7.2). Line search is performed to find the constant $c > 0$ that yields an adversarial example at minimum distance: in other words, the optimization problem is solved for multiple values of $c$, adaptively updating $c$ using bisection search or any other method for one-dimensional optimization.

In general, L-BFGS is computationally very demanding; therefore, fast, although suboptimum, attack algorithms have also been proposed.

### 7.2.2 Fast Gradient Sign Method (FGSM) and Iterative FGSM (I-FGSM)

The Fast Gradient Sign Method (FGSM) was originally proposed in [14], as a fast, suboptimum, attack method.

Given an image $X$, FGSM sets

$$X' = X + \varepsilon(\max(X) - \min(X)) \cdot \mathrm{sign}(\nabla_X J_\theta(X, y)) \tag{7.3}$$

where $\varepsilon$ is the (normalized) strength of the attack which has to be sufficiently small so as to be undetectable, and strong enough to achieve misclassification. Intuitively, for each pixel, the fast gradient sign method uses the gradient of the loss function to determine in which direction the pixel's intensity should be changed; then, it shifts all pixels simultaneously. The fast gradient sign attack was designed to be a fast attack, computationally much more efficient than L-BFGS, yet it does not produce a close-to-minimal adversarial perturbation.

(another difference with respect to the L-BFGS is that optimized for the $L^\infty$ distance metric instead of the $L_2$.)

The one-shot attack in (7.3) often fails to get the adversarial images. A less suboptimum scheme can be obtained by considering the refined *iterative* version of the attack: specifically, at each iteration $i+1$, the adversarial perturbation is obtained as detailed above and the image is updated as

$$X_{i+1} = X_i + \varepsilon(\max(X_i) - \min(X_i)) \cdot \mathrm{sign}(\nabla_X J_\theta(X_i, y)) \qquad (7.4)$$

for some *small* strength $\varepsilon$ (hence, each time, the attack makes a small step in the direction of the gradient sign).

In practical implementations, the iterative algorithm is applied for a maximum number of steps (the process stops when an adversarial image is obtained, prior that the prescribed maximum number of steps is reached).

### 7.2.3 Jacobian-based Saliency Map Attack (JSMA)

Papernot et al. introduced an attack optimized under the $L_0$ distance [119], known as the Jacobian-based Saliency Map Attack (JSMA). Roughly speaking, JSMA consists of a greedy iterative procedure which relies on forward propagation to compute, at each iteration, a *saliency map*, indicating the pixels that contribute most to the classification (specifically, a large value in this map indicates that changing that pixel yields a significant increase of the likelihood of the wrong class). The pixels are then modified based on this map, one at the time, by a relative amount $\theta$, $\theta < 1$ ($\theta$ is relative to the range of the values of the image, the pixel modification being $\theta \cdot (\max(X_i) - \min(X_i))$). The procedure ends when one of the following conditions occurs: the attacker succeeds in changing the classification result, hence getting an adversarial image $X_{adv}$ such that $l(X_{adv}) \neq y$; too many pixels are modified by the attack (that is, an adversarial image cannot be found for a given maximum $L_0$ distortion); the prescribed maximum number of iterations is reached.

### 7.2.4 Projected Gradient Descent (PGD)

The Projected Gradient Descent (PGD) attack, proposed by [185], cares about finding the perturbation that maximizes the loss function under some restrictions regarding the $L^\infty$ distortion.

At each iteration $i + 1$, the image is first updated according to some rule getting $X_{i+1}$, then, the image is projected onto the space of the images having a constrained $L^\infty$ distortion from the original image, with the maximum distortion set to some value $\alpha$, that is $X_{i+1} = \Pi_{X+\alpha}(X_{i+1})$ (where $\Pi$ denotes the projection operator).

When FGSM is considered for the updating (which is the case in practice), PGD is a multi-step extension of the FGSM attack, similar to the I-FGSM. Following the (simplified) attack procedure proposed by [186], PDG is implemented as follows: the basic gradient sign attack is applied multiple times with a small step size or strength $\varepsilon$ ($\varepsilon < \alpha$); after every step, the pixel values of every intermediate results are clipped to ensure that the adversarial image remains in the $\alpha$-neighbourhood of the original image. Formally, at each iteration $i + 1$, the adversarial perturbation is refined by computing

$$[X_{i+1}]_{r,c} = \text{clip}([X_{i+1}]_{r,c}, \{-\alpha, +\alpha\}), \tag{7.5}$$

for pixel $(r, c)$, where $X_{i+1}$ takes the expression in (7.4).

Due to clipping, this attack may result in a highly suboptimal solution in some cases. A binary search can be performed over $\varepsilon$ and $\alpha$ to optimize the choice of the hyperparameters.

In [186], the PDG attack applied as described above is called Basic Iterative Method.

## 7.3 Methodology

In order to evaluate the factors that influence the transferability of adversarial attacks against CNN-based detection of image processing operators, we considered two different kinds of attacks, two detection tasks solved by relying on two different networks, and three sources of mismatch between the network used to create the adversarial attack (hereafter referred to as Source Network - SN) and the one the attack should be transferred to (hereafter referred to as Target Network - TN). We analysed separately the effect of different sources of mismatches, namely, training data mismatch and network architecture mismatch, on the transferability of the attacks.

In particular, we considered the cases of two different networks trained on the same dataset and the case of a single network trained on different datasets. With reference to the terminology established in [13], we refer to the first type of transferability as *cross-model transferability* and to the second as *cross-training transferability*. We also considered the case of two different networks trained on different datasets (*cross-model-and-training transferability*). The combination of the above factors resulted in an extensive campaign of experiments whose results will be discussed in Section 7.4.

### 7.3.1 Adversarial attacks

In our experiments, the adversarial examples were built by relying on the FGSM, the JSMA algorithm, and the L-BFGS algorithm, described above. The Foolbox toolbox [187] was used to implement them. The reason why we considered in particular the FGSM and JSMA attacks is the following: being suboptimal attack methods (that attempt to induce a misclassification while keeping the distortion limited), FGSM and JSMA are expected to be more robust to a mismatch, hence more transferable, with respect to other more-close to the optimum gradient-descent attacks, and in particular, the L-BFGS attack, that looks for the 'minimum' adversarial perturbation causing misclassification (i.e., minimizes the distortion). This claim is also supported by the results we got in Section 7.4.

For the FGSM, we considered the refined iterative version I-FGSM. As mentioned before (Section 7.2.2), the algorithm is applied iteratively for a maximum number of steps $S$. Moreover, several values of $\varepsilon$ are considered, i.e. $\varepsilon \in E$; then, the value which minimizes the distortion of the final attacked image with respect to the original one is selected, for the given maximum number of iterations of the algorithm $S$. In the I-FGSM implementation in Foolbox, $\varepsilon$ corresponds to the normalized strength factor (same convention of the formalism above (7.4)).

With regard to the JSMA algorithm, in addition to the relative amount $\theta$ of pixel modification at each iteration, another important parameter is the maximum number of times $T$ the same pixel can be modified. We do not consider any limit on the maximum number of iterations; then, the procedure ends when the attacker succeeds or the pixels are modified by a too large amount (i.e., the number of modifications reaches the maximum prescribed number for all pixels).

Both the I-FGSM and the JSMA algorithms produce a real-valued attacked image. While in some cases we can assume that the attacked image is used as is, in most applications image pixels must be mapped back into the integer domain before being fed to the CNN. This may result in a loss of effectiveness of the attack, since some of the subtle changes introduced by the attack are deleted when pixels are rounded (or truncated) to integer values.

### 7.3.2 Datasets

In order to evaluate the transferability of the attacks when the SN and the TN are trained on different datasets, we considered the RAISE-2K (R) dataset and the VISION (V) dataset.

For our experiments, all the 2000 uncompressed, camera-native, images (.tiff) were taken from the RAISE-2K dataset. The same number of images were taken from the VISION dataset. To get similar resolution images for the two datasets, we only selected the mobile devices for which the resolution was not very different from that of the images of RAISE. Specifically, the sizes of the images we considered range from a minimum of $2336 \times 4160$ up to $3480 \times 4640$. The images from VISION dataset are in JPEG format. In order to reduce the possible impact of compression artefact, we selected images for which the JPEG Quality Factor is larger than 97. The images from both $R$ and $V$ datasets were split into training (and validation) set and test set, and then processed to produce the images for the manipulated class, namely, median and resizing. For all our tests we considered one-channel images, then all the images from $R$ and $V$ were converted to gray-scale.

### 7.3.3 Networks

In our experiments, we considered two different detection tasks, namely the detection of image resizing (downsampling by a 0.8 factor) and median filtering (by a $5 \times 5$ window). To cope with them, we built several networks generally indicated as $N_{\mathrm{ar}}^{\mathrm{tr}}(\mathrm{task})$, where "ar" indicates the architecture of the network, "tr" $\in \{\mathrm{R}, \mathrm{V}\}$ the dataset used for training and "task" $\in \{\mathrm{med}, \mathrm{res}\}$ the detection task ("med" indicating median filtering and "res" resizing).

With regard to the architectures, we considered the network in [8] (recently extended in [74]), hereafter referred to as BSnet ("ar" $=$ BS), and the one presented in Chapter 5, Section 5.3, hereafter denoted as GCnet (”ar” = GC). BSnet, originally proposed for image manipulation detection and classification, consists of 3 convolutional layers, 3 max-pooling layers and 3 fully-connected layers. Residual-based features are extracted by constraining the filters of the first layer (with $5 \times 5$ receptive field), by enforcing a high-pass nature of the filters (see [74] for more details). For the second and third convolutional layers the filter size is set to $7 \times 7$ and $5 \times 5$ respectively, and the stride is set to 2. For the max-pooling, a kernel size $3 \times 3$ is used with stride 2. GCnet is significantly deeper than BSnet, consisting of 9 convolutional layers. The network has only 2 max-pooling layers and one fully-connected layer. A kernel size of $3 \times 3$ and stride 1 was used for all the convolutional layers. Max-pooling is applied with kernel size $2 \times 2$ and stride 2. The number of parameters is then reduced by halving the number of feature maps in the final convolutional layer, and considering just one fully-connected layer. The reader may refer to Section 5.3 for more details on the GCnet architecture.

### 7.3.4 Experiments

The experimental campaign was designed in such a way to highlight attack transferability in a wide variety of settings. Experiments have been split into three categories according to the type of mismatch between the SN and the TN. We started studying cross-training transferability, according to which SN and TN share the same architecture, but are trained on different datasets. Then we passed to analyse cross-model transferability, in which different network architectures are trained on the same dataset. Eventually, we passed to cross-model-and-training transferability according to which the SN and the TN share neither the architecture nor the training data. All the tests have been repeated for both resizing and median filtering detection. For sake of simplicity we did not consider all possible combinations, however, the amount of experiments we carried out is sufficient to draw a number of significant conclusions. In particular, the experiments for the cross-training transferability are carried out by considering only BSnet as the SN, trained on $R$ i.e., $SN = N^{\text{R}}_{\text{BS}}$ (in this case $TN = N^{\text{V}}_{\text{BS}}$) and on $V$ i.e., $SN = N^{\text{R}}_{\text{BS}}$ ($TN = N^{\text{R}}_{\text{BS}}$). For the experiments on the cross-model transferability, BSnet is taken as SN, GCnet as TN, both trained on $R$ i.e., $SN = N^{\text{R}}_{\text{BS}}$ and $TN = N^{\text{R}}_{\text{GC}}$. Finally, for the cross-model-and-training case, we set $SN = N_{\mathrm{BS}}^{\mathrm{V}}$ and $TN = N_{\mathrm{GC}}^{\mathrm{R}}$. Table 7.1 summarize the notation for the 6 networks, along with the information about the architecture, training dataset and task.

Table 7.1: The 6 trained network models considered in the experiments.

| **Network** | **Task** | **Architecture** | **Training Dataset** |
|---|---|---|---|
| $N_{\mathrm{BS}}^{\mathrm{R}}$(res) | resize | BSnet | RAISE |
| $N_{\mathrm{BS}}^{\mathrm{V}}$(res) | resize | BSnet | VISION |
| $N_{\mathrm{BS}}^{\mathrm{R}}$(med) | median filtering | BSnet | RAISE |
| $N_{\mathrm{BS}}^{\mathrm{V}}$(med) | median filtering | BSnet | VISION |
| $N_{\mathrm{GC}}^{\mathrm{R}}$(res) | resize | GCnet | RAISE |
| $N_{\mathrm{GC}}^{\mathrm{R}}$(med) | median filtering | GCnet | RAISE |

With regard to the attacks, for the I-FGSM case, the number of steps $S$ is fixed to 10 (default). The best strength is searched in the range $E = [0 : \varepsilon_s : 0.1]$, where $\varepsilon_s$ is the search step size, which also corresponds to the minimum (normalized) strength $\varepsilon$ considered. Setting a larger $\varepsilon_s$ generally corresponds to consider a stronger attack. In our experiments, we considered $\varepsilon_s = 0.001$ and 0.01, for which the average PSNR remains above 40 dB. For the JSMA, $T$ is set to 7. The relative modification per pixel $\theta$ is set to 0.01 and 0.1, the second case corresponding to a stronger attack. We did not consider $\theta$ values larger than 0.1, since above this value the maximum pixel distortion introduced by the attack starts becoming too large ($> 70$). For the BFGS, we considered the default parameter for $\varepsilon$. Eventually, we repeated all the experiments by rounding the output of the attack to integer values.

## 7.4 Results and Discussion

In this section we discuss the results of the experiments we have carried out. We will focus on the floating point version of the attacks, being this case more favorable to the attacker. For completeness, we also report the results for the integer-valued case, that is, when the pixel values of the attacked images are rounded to integers. According to our results, integer rounding does not have a big impact on the transferability of the attacks. Rather it influences the effectiveness of the attack on the SN itself, as already reported in several studies e.g. [116, 124].

To build our models $N_{\text{BS}}^{\text{R}}$ and $N_{\text{BS}}^{\text{V}}$ (for both detection tasks), we considered 200.000 patches per class for training (and validation) and 10000 for testing. In order to use all the images in the datasets $R$ and $V$, a maximum number of 100 patches is selected (randomly) for each image. A number of 30 training epochs was considered (as in [8]). For the deeper models $N_{\text{GC}}^{\text{R}}$ for both the "med" and "res" task, we used $10^6$ patches for training, $10^5$ for validation, and $5 * 10^4$ for testing. To reach these numbers, all the image patches were selected from all the images. By following the original setting (see Section 5.3), the number of training epochs is set to 3. The input patch size is set to $128 \times 128$. For training both BSnet and GCnet, the Adam solver is used with learning rate $10^{-4}$ and momentum 0.99. The batch size for training is set to 32 images, the test batch size to 100. Table 7.2 shows the accuracies achieved by the BSnet and GCnet in the absence of attacks in the various cases.

Table 7.2: Accuracies of the trained models in the absence of attacks in the various cases.

| **Network** | $N_{\text{BS}}^{\text{R}}$(med) | $N_{\text{BS}}^{\text{V}}$(med) | $N_{\text{BS}}^{\text{R}}$(res) | $N_{\text{BS}}^{\text{V}}$(res) | $N_{\text{GC}}^{\text{R}}$(med) | $N_{\text{GC}}^{\text{R}}$(res) |
|---|---|---|---|---|---|---|
| **Accuracy** | 98.1% | 99.5% | 97.5% | 96.6% | 98.4% | 98.5% |

In the next section, we discuss the performance of the models in the presence of attacks, in the matched and mismatched cases. In counter-forensic applications, it is reasonable to assume that the attacks is only in one direction, since the attacker wants to pass off a manipulated image as an original one. Therefore, in our experiments, we only attack images from the manipulated class. In all the cases, the performance of the attack are assessed on 500 patches, obtained by attacking a subset of the patches from the corresponding test set in each case. Obviously, we attack only images for which the classification of the network is correct. An attack is declared successful when it is able to switch the network decision, i.e., when the manipulated image is labeled as original after the attack (the output soft score for the original class becomes larger than 0.5).

Table 7.3 shows the accuracies achieved by the networks in the absence of attacks on the manipulated class in various cases.

Table 7.3: Accuracies of the trained models in absence of attacks on the manipulated class ($H_1$) in various cases.

| **Network** | $N^{\mathrm{R}}_{\mathrm{BS}}$(med) | $N^{\mathrm{V}}_{\mathrm{BS}}$(med) | $N^{\mathrm{R}}_{\mathrm{BS}}$(res) | $N^{\mathrm{V}}_{\mathrm{BS}}$(res) | $N^{\mathrm{R}}_{\mathrm{GC}}$(med) | $N^{\mathrm{R}}_{\mathrm{GC}}$(res) |
|---|---|---|---|---|---|---|
| **Accuracy** | 98.2% | 100% | 97.6% | 97.8% | 100% | 99.6% |

### 7.4.1 Cross-training transferability

As detailed in Section 7.3.4, these experiments were carried out by considering only the BS architecture. The results we got are reported in Table 7.4 for resizing task and in Table 7.5 for median filtering task. For each attack type, the table report the PSNR, $L_1$ distortion and maximum absolute distortion, averaged on all the images successfully attacked in the matched case, i.e., successfully fooling SN (see the last paragraph above, immediately before Section 7.4.1). The attack success rate (ASR) with respect to SN, TN, and ASR with respect to TN on integer-valued attacked samples is reported in the last three columns. As we can see, the attacks are generally non-transferable and the images attacked using SN are not able to deceive the TN. More specifically, with the FGSM attack, the adversarial examples can be transferred in a significant number of cases only when the larger strength is considered ($\varepsilon_s = 0.01$) and the SN corresponds to $N^{\mathrm{R}}_{\mathrm{BS}}$(res) and $N^{\mathrm{R}}_{\mathrm{BS}}$(med) (attack success rate 0.692 and 0.845 respectively) and to $N^{\mathrm{V}}_{\mathrm{BS}}$(res) (attack success rate 0.941). For the JSMA case, the attack can be transferred only when SN is $N^{\mathrm{R}}_{\mathrm{BS}}$(res) and strong attack with $\theta = 0.1$ is considered, with success rate 0.782. Furthermore, we observe that the JSMA is never transferable when the VISION dataset is used to train the SN. For the BFGS case, we observe that the attack is not transferable, either with RAISE or VISION for the resize and median tasks (the largest ASR that can be obtained on TN being 20-30%). It is also interesting to observe that, for a given detection task, the transferability is not symmetric with respect to the datasets used for training. This suggests that, in forensic applications, the features learned by the network may also be affected in some way and up to some extent by the underlying dataset. If the model targeted by the attack (SN) has for some reasons a more intricate decision boundary, the attack could generalize less to other models, thus being less transferable.

Table 7.4: Experimental results for Cross Training (resizing task). Transferable attacks are highlighted in bold.

| SN | TN | Att type | PSNR | L1 dist | max dist | ASR SN | ASR TN | ASR TN (Int) |
|---|---|---|---|---|---|---|---|---|
| $N^{\mathrm{R}}_{\mathrm{BS}}$(res) | $N^{\mathrm{V}}_{\mathrm{BS}}$(res) | I-FGSM<br>$\varepsilon_s = 0.01$ | 40.02 | 2.53 | 2.55 | **1.000** | **0.692** | **0.711** |
| $N^{\mathrm{R}}_{\mathrm{BS}}$(res) | $N^{\mathrm{V}}_{\mathrm{BS}}$(res) | I-FGSM<br>$\varepsilon_s = 0.001$ | 58.46 | 0.26 | 0.27 | 1.000 | 0.0491 | 0.019 |
| $N^{\mathrm{R}}_{\mathrm{BS}}$(res) | $N^{\mathrm{V}}_{\mathrm{BS}}$(res) | JSMA<br>$\theta = 0.1$ | 46.04 | 0.07 | 58.32 | **1.000** | **0.782** | **0.786** |
| $N^{\mathrm{R}}_{\mathrm{BS}}$(res) | $N^{\mathrm{V}}_{\mathrm{BS}}$(res) | JSMA<br>$\theta = 0.01$ | 54.99 | 0.04 | 15.09 | 0.991 | 0.115 | 0.136 |
| $N^{\mathrm{R}}_{\mathrm{BS}}$(res) | $N^{\mathrm{V}}_{\mathrm{BS}}$(res) | BFGS<br>$\varepsilon = 0.1$ | 57.81 | 0.25 | 2.13 | 1.000 | 0.261 | 0.169 |
| $N^{\mathrm{V}}_{\mathrm{BS}}$(res) | $N^{\mathrm{R}}_{\mathrm{BS}}$(res) | I-FGSM<br>$\varepsilon_s = 0.01$ | 40.03 | 2.53 | 2.55 | 1.000 | 0.002 | 0.002 |
| $N^{\mathrm{V}}_{\mathrm{BS}}$(res) | $N^{\mathrm{R}}_{\mathrm{BS}}$(res) | I-FGSM<br>$\varepsilon_s = 0.001$ | 59.64 | 0.26 | 0.27 | 1.000 | 0.000 | 0.000 |
| $N^{\mathrm{V}}_{\mathrm{BS}}$(res) | $N^{\mathrm{R}}_{\mathrm{BS}}$(res) | JSMA<br>$\theta = 0.1$ | 50.55 | 0.01 | 69.42 | 0.989 | 0.000 | 0.000 |
| $N^{\mathrm{V}}_{\mathrm{BS}}$(res) | $N^{\mathrm{R}}_{\mathrm{BS}}$(res) | JSMA<br>$\theta = 0.01$ | 57.78 | 0.01 | 17.06 | 0.979 | 0.000 | 0.000 |
| $N^{\mathrm{V}}_{\mathrm{BS}}$(res) | $N^{\mathrm{R}}_{\mathrm{BS}}$(res) | BFGS<br>$\varepsilon = 0.1$ | 57.62 | 0.25 | 2.26 | 1.000 | 0.002 | 0.002 |

### 7.4.2 Cross-model transferability

In this case, the experiments were carried out by considering only the $R$ dataset and using the BS architecture for the SN. The results we have got are reported in Table 7.6 for resizing task and Table 7.7 for median filtering task. The experiments show the lack of transferability with respect to a mismatch in the network model. The only exception is for the "med" case, in which case the stronger attack (with $\varepsilon_s = 0.01$) is transferable 82.5% of the times. However, it is worth stressing that, when the FGSM is applied with such a strength,

Table 7.5: Experimental results for Cross Training (median filtering task). Transferable attacks are highlighted in bold.

| SN | TN | Att<br>type | PSNR | L1<br>dist | max<br>dist | ASR<br>SN | ASR<br>TN | ASR<br>TN (Int) |
|---|---|---|---|---|---|---|---|---|
| $N_{\text{BS}}^{\text{R}}$(med) | $N_{\text{BS}}^{\text{V}}$(med) | I-FGSM<br>$\varepsilon_s = 0.01$ | 40.03 | 2.53 | 2.55 | **1.000** | **0.845** | **0.904** |
| $N_{\text{BS}}^{\text{R}}$(med) | $N_{\text{BS}}^{\text{V}}$(med) | I-FGSM<br>$\varepsilon_s = 0.001$ | 59.67 | 0.26 | 0.27 | 1.000 | 0.045 | 0.000 |
| $N_{\text{BS}}^{\text{R}}$(med) | $N_{\text{BS}}^{\text{V}}$(med) | JSMA<br>$\theta = 0.1$ | 49.64 | 0.03 | 38.11 | 1.000 | 0.012 | 0.012 |
| $N_{\text{BS}}^{\text{R}}$(med) | $N_{\text{BS}}^{\text{V}}$(med) | JSMA<br>$\theta = 0.01$ | 58.47 | 0.02 | 14.05 | 0.984 | 0.002 | 0.002 |
| $N_{\text{BS}}^{\text{R}}$(med) | $N_{\text{BS}}^{\text{V}}$(med) | BFGS<br>$\varepsilon = 0.1$ | 56.55 | 0.28 | 2.53 | 1.000 | 0.276 | 0.246 |
| $N_{\text{BS}}^{\text{V}}$(med) | $N_{\text{BS}}^{\text{R}}$(med) | I-FGSM<br>$\varepsilon_s = 0.01$ | 40.04 | 2.53 | 2.55 | **1.000** | **0.941** | **0.953** |
| $N_{\text{BS}}^{\text{V}}$(med) | $N_{\text{BS}}^{\text{R}}$(med) | I-FGSM<br>$\varepsilon_s = 0.001$ | 59.94 | 0.25 | 0.25 | 1.000 | 0.077 | 0.000 |
| $N_{\text{BS}}^{\text{V}}$(med) | $N_{\text{BS}}^{\text{R}}$(med) | JSMA<br>$\theta = 0.1$ | 49.55 | 0.03 | 32.09 | 1.000 | 0.010 | 0.010 |
| $N_{\text{BS}}^{\text{V}}$(med) | $N_{\text{BS}}^{\text{R}}$(med) | JSMA<br>$\theta = 0.01$ | 58.13 | 0.01 | 14.08 | 0.988 | 0.008 | 0.010 |
| $N_{\text{BS}}^{\text{V}}$(med) | $N_{\text{BS}}^{\text{R}}$(med) | BFGS<br>$\varepsilon = 0.1$ | 56.66 | 0.27 | 2.54 | 1.000 | 0.300 | 0.296 |

although the PSNR is not very low (40.03 dB), the average $L_1$ distortion is around 2.5 (a similar value is attained by the maximum absolute distortion). With such values of $L_1$, the visual quality of the FGSM attacked images is impaired and peculiar visual artifacts appears, especially in relatively uniform image patches.

The fact that the lack of transferability is even stronger in the ”res” case than in the ”med” case can be probably justified by the ease of the median filtering detection task (even because the median filtering is performed with a rather large window size), compared to the resize. Therefore, we might expect that in the case of ”med” similar peculiar features are learned by the shallow and deeper network, hence facilitating the transferability of the attacks.

Table 7.6: Experimental results for Cross Model (resizing task).

| SN | TN | Att type | PSNR | L1 dist | max dist | ASR SN | ASR TN | ASR TN (Int) |
|---|---|---|---|---|---|---|---|---|
| $N^{\mathrm{R}}_{\mathrm{BS}}$(res) | $N^{\mathrm{R}}_{\mathrm{GC}}$(res) | I-FGSM<br>$\varepsilon_s = 0.01$ | 40.02 | 2.53 | 2.55 | 1.000 | 0.002 | 0.004 |
| $N^{\mathrm{R}}_{\mathrm{BS}}$(res) | $N^{\mathrm{R}}_{\mathrm{GC}}$(res) | I-FGSM<br>$\varepsilon_s = 0.001$ | 58.48 | 0.31 | 0.33 | 1.000 | 0.002 | 0.000 |
| $N^{\mathrm{R}}_{\mathrm{BS}}$(res) | $N^{\mathrm{R}}_{\mathrm{GC}}$(res) | JSMA<br>$\theta = 0.1$ | 46.09 | 0.07 | 57.88 | 1.000 | 0.016 | 0.010 |
| $N^{\mathrm{R}}_{\mathrm{BS}}$(res) | $N^{\mathrm{R}}_{\mathrm{GC}}$(res) | JSMA<br>$\theta = 0.01$ | 54.98 | 0.04 | 15.14 | 0.992 | 0.006 | 0.008 |
| $N^{\mathrm{R}}_{\mathrm{GC}}$(res) | $N^{\mathrm{R}}_{\mathrm{BS}}$(res) | JSMA<br>$\theta = 0.1$ | 46.34 | 0.20 | 26.81 | 1.000 | 0.004 | 0.004 |
| $N^{\mathrm{R}}_{\mathrm{BS}}$(res) | $N^{\mathrm{R}}_{\mathrm{GC}}$(res) | BFGS<br>$\varepsilon = 0.1$ | 57.80 | 0.25 | 2.13 | 1.000 | 0.061 | 0.063 |

Table 7.7: Experimental results for Cross Model (median filtering task). Transferable attacks are highlighted in bold.

| SN | TN | Att type | PSNR | L1 dist | max dist | ASR SN | ASR TN | ASR TN (Int) |
|---|---|---|---|---|---|---|---|---|
| $N^{\mathrm{R}}_{\mathrm{BS}}$(med) | $N^{\mathrm{R}}_{\mathrm{GC}}$(med) | I-FGSM<br>$\varepsilon_s = 0.01$ | 40.03 | 2.53 | 2.55 | **1.000** | **0.825** | **0.877** |
| $N^{\mathrm{R}}_{\mathrm{BS}}$(med) | $N^{\mathrm{R}}_{\mathrm{GC}}$(med) | I-FGSM<br>$\varepsilon_s = 0.001$ | 59.67 | 0.26 | 0.27 | 1.000 | 0.181 | 0.004 |
| $N^{\mathrm{R}}_{\mathrm{BS}}$(med) | $N^{\mathrm{R}}_{\mathrm{GC}}$(med) | JSMA<br>$\theta = 0.1$ | 49.64 | 0.03 | 38.11 | 1.000 | 0.010 | 0.010 |
| $N^{\mathrm{R}}_{\mathrm{BS}}$(med) | $N^{\mathrm{R}}_{\mathrm{GC}}$(med) | JSMA<br>$\theta = 0.01$ | 58.47 | 0.02 | 14.05 | 0.984 | 0.016 | 0.018 |
| $N^{\mathrm{R}}_{\mathrm{GC}}$(med) | $N^{\mathrm{R}}_{\mathrm{BS}}$(med) | JSMA<br>$\theta = 0.1$ | 47.52 | 0.06 | 27.76 | 1.000 | 0.022 | 0.022 |
| $N^{\mathrm{R}}_{\mathrm{BS}}$(med) | $N^{\mathrm{R}}_{\mathrm{GC}}$(med) | BFGS<br>$\varepsilon = 0.1$ | 56.55 | 0.28 | 2.53 | 1.000 | 0.272 | 0.272 |

We also performed some experiments by considering GCnet as SN, that is considering $SN = N^{\mathrm{R}}_{\mathrm{GC}}(\mathrm{med})$ and $TN = N^{\mathrm{R}}_{\mathrm{BS}}(\mathrm{med})$ and $SN = N^{\mathrm{R}}_{\mathrm{GC}}(\mathrm{res})$

and $TN = N^{\mathrm{R}}_{\mathrm{BS}}(\mathrm{res})$. The overall degree of transferability of the attacks that can be obtained is similar to the case in which BSnet is used as SN. However, differently from the previous case, in this case there are many images for which the network can not be attacked (then, the ASR on SN is lower). This is due to the so called gradient-vanishing problem [44], that is, for some images, the gradient is so (vanishingly) small, that the images can not be updated during the attack (a descent direction can not be found). In these cases, the attack fails again SN. This reveals a clear asymmetry of the attack (and then as a consequence of the transferability) with respect to the architecture used to build the SN. Notably, deeper networks are more prone to the gradient-vanishing problem. The results we got from our experiments are reported in the Table 7.8.

Table 7.8: Experimental results for Cross Model (median filtering and resizing task) by considering GCnet as SN.

| SN | TN | Att type | PSNR | ASR SN | ASR TN |
|---|---|---|---|---|---|
| $N^{\mathrm{R}}_{\mathrm{GC}}(\mathrm{med})$ | $N^{\mathrm{R}}_{\mathrm{BS}}(\mathrm{med})$ | BFGS | 66.38 | 0.902 | 0.015 |
| $N^{\mathrm{R}}_{\mathrm{GC}}(\mathrm{res})$ | $N^{\mathrm{R}}_{\mathrm{BS}}(\mathrm{res})$ | BFGS | 70.34 | 0.576 | 0.017 |
| $N^{\mathrm{R}}_{\mathrm{GC}}(\mathrm{med})$ | $N^{\mathrm{R}}_{\mathrm{BS}}(\mathrm{med})$ | I-FGSM | 57.35 | 0.902 | 0.319 |
| $N^{\mathrm{R}}_{\mathrm{GC}}(\mathrm{res})$ | $N^{\mathrm{R}}_{\mathrm{BS}}(\mathrm{res})$ | I-FGSM | 58.86 | **0**.576 | 0.007 |

### 7.4.3 Cross-model-and-training transferability

In this case, the experiments were carried out by considering the BS architecture trained on the $V$ dataset as the SN, and the GC architecture trained on the $R$ dataset as the TN. Similar results can be obtained by combining architecture and dataset in the other way round. The results we have got are reported in Table 7.9 for resizing task and Table 7.10 for median filtering task. Quite expectedly, the table shows that the transferability of the attacks in this case decreases further and the attack success rate is below 0.01 in all the cases but for the case of FGSM with $\varepsilon_s = 0.01$, for which a success rate of 0.796 can still be achieved.

As a general behavior, according to our tests, for all the three types of mismatch considered, attacks obtained by JSMA are less transferable than those produced by FGSM. A possible motivation can be the following: since very few pixels are modified by JSMA (those which the network is more sensitive to), it tends to overfit more the attacked model. We also observe that, with JSMA, the average output scores returned by SN and TN on the successfully attacked and transferred samples, are often very different (while with FGSM they are more similar). We also observe that, while with JSMA the SN scores for these samples are in the range $[0.5, 0.6]$, with FGSM, such output scores are always much larger than 0.5 (often $> 0.9$), both with SN and TN.

Table 7.9: Experimental results for Cross Model and Training (resizing task).

| SN | TN | Att type | PSNR | L1 dist | max dist | ASR SN | ASR TN | ASR TN (Int) |
|---|---|---|---|---|---|---|---|---|
| $N_{\text{BS}}^{\text{V}}$(res) | $N_{\text{GC}}^{\text{R}}$(res) | I-FGSM, $\varepsilon_s = 0.01$ | 40.03 | 2.53 | 2.55 | 1.000 | 0.004 | 0.004 |
| $N_{\text{BS}}^{\text{V}}$(res) | $N_{\text{GC}}^{\text{R}}$(res) | I-FGSM, $\varepsilon_s = 0.001$ | 59.57 | 0.27 | 0.27 | 1.000 | 0.002 | 0.000 |
| $N_{\text{BS}}^{\text{V}}$(res) | $N_{\text{GC}}^{\text{R}}$(res) | JSMA, $\theta = 0.1$ | 50.20 | 0.02 | 70.87 | 1.000 | 0.000 | 0.000 |
| $N_{\text{BS}}^{\text{V}}$(res) | $N_{\text{GC}}^{\text{R}}$(res) | JSMA, $\theta = 0.01$ | 57.40 | 0.01 | 17.16 | 0.992 | 0.000 | 0.000 |
| $N_{\text{GC}}^{\text{R}}$(res) | $N_{\text{BS}}^{\text{V}}$(res) | JSMA, $\theta = 0.1$ | 45.22 | 0.09 | 26.96 | 1.000 | 0.171 | 0.173 |
| $N_{\text{BS}}^{\text{V}}$(res) | $N_{\text{GC}}^{\text{R}}$(res) | BFGS, $\varepsilon = 0.1$ | 57.62 | 0.25 | 2.26 | 1.000 | 0.0103 | 0.002 |

## 7.5 Concluding Remarks

By focusing on two manipulation detection tasks, we investigated the transferability of adversarial examples in an image forensics scenario. We run tests by considering two well known attack methodologies and several sources of mismatch. Our tests show that adversarial examples are generally non-transferable, in contrast to what happens in typical pattern recognition applications. This states an important result, since the lack of transferability

Table 7.10: Experimental results for Cross Model and Training (median filtering task).

| SN | TN | Att type | PSNR | L1 dist | max dist | ASR SN | ASR TN | ASR TN (Int) |
|---|---|---|---|---|---|---|---|---|
| $N_{\mathrm{BS}}^{\mathrm{V}}$(med) | $N_{\mathrm{GC}}^{\mathrm{R}}$(med) | I-FGSM, $\varepsilon_s = 0.01$ | 40.04 | 2.53 | 2.55 | **1.000** | **0.796** | **0.830** |
| $N_{\mathrm{BS}}^{\mathrm{V}}$(med) | $N_{\mathrm{GC}}^{\mathrm{R}}$(med) | I-FGSM, $\varepsilon_s = 0.001$ | 59.91 | 0.25 | 0.26 | 1.000 | 0.008 | 0.000 |
| $N_{\mathrm{BS}}^{\mathrm{V}}$(med) | $N_{\mathrm{GC}}^{\mathrm{R}}$(med) | JSMA, $\theta = 0.1$ | 49.56 | 0.03 | 31.83 | 1.000 | 0.008 | 0.008 |
| $N_{\mathrm{BS}}^{\mathrm{V}}$(med) | $N_{\mathrm{GC}}^{\mathrm{R}}$(med) | JSMA, $\theta = 0.01$ | 58.06 | 0.01 | 14.18 | 0.990 | 0.012 | 0.012 |
| $N_{\mathrm{GC}}^{\mathrm{R}}$(med) | $N_{\mathrm{BS}}^{\mathrm{V}}$(med) | JSMA, $\theta = 0.1$ | 47.61 | 0.06 | 27.70 | 0.997 | 0.022 | 0.060 |
| $N_{\mathrm{BS}}^{\mathrm{V}}$(med) | $N_{\mathrm{GC}}^{\mathrm{R}}$(med) | BFGS, $\varepsilon = 0.1$ | 56.66 | 0.27 | 2.53 | 1.000 | 0.048 | 0.062 |

can be exploited by the forensic analyst to make the attack more difficult. For instance, a LK scenario can be enforced in some way to combat adversarial examples, as done for the approaches based on standard ML techniques.

To further investigate this important result, different sources of mismatch between the SN and the TN should also be considered and analyzed. As an example, we may wonder if a mismatch in the training procedure is enough to prevent transferability. Moreover, the reason why image-forensic networks are less prone to attack transfers should be understood.

On the attacker's hand, as a result of our analysis, research is needed to understand if and how the transferability can be improved by increasing the strength of the attack in such a way to enter more inside the target region.

It is worth pointing that, increasing the strength of the attacks in order to enter more inside the target region is not an easy task, given the complexity of the decision boundary learnt by the CNNs. The amount of distortion introduced in the image by the attack, or the value assumed by the decision function, in fact, represent only an inaccurate proxy for the attack strength, since controlling the amount of distortion (e.g., by letting the gradient descent continue until a limit PSNR is reached for the attack), or setting a safe margin on the decision function for the attack, do not necessarily result in a stronger attack.

Further results and considerations pertaining the attack transferability will follow from the analysis of the effectiveness of the randomization-based defence against adversarial examples carried out in the following chapter.

# Chapter 8

# Effectiveness of Random Deep Feature Selection for Securing Image Manipulation Detectors Against Adversarial Examples

> *"All objects in the universe are unique. No two things that happen by chance ever happen in exactly the same way. No two things are ever constructed or manufactured in exactly the same way. No two things wear in exactly the same way. No two things ever break in exactly the same way."*
>
> Joe Nickell
>
> *"We can all see, but can you observe?"*
>
> A.D. Garrett, Everyone Lies

The purpose of this chapter is to investigate if the random feature selection approach, proposed in the literature to improve the robustness of forensic detectors to targeted attacks for general model-based and standard ML-based forensics, can be applied to detectors based on deep learning. In particular, we study the transferability of adversarial examples targeting an original CNN image manipulation detector to other detectors (a fully connected neural network and a linear SVM) that rely on a random subset of the features extracted from the flatten layer of the original network. The research is conducted by considering three image manipulation detection tasks (resizing, median filtering and contrast enhancement), two original network architectures (the same BSnet and GCnet considered in Chapter 7) and three classes of attacks.

This chapter is organised as follows: we first briefly explain the reasons that pushed us to study the random feature selection approach in Section 8.1. Then, the general proposed Random Deep Feature Selection (RDFS) scheme is presented in Section 8.2. The methodology we followed for our experiments is described in Section 8.3. The results we got are reported and discussed in Section 8.4.

## 8.1 Motivation

As we discussed in Section 3.3 of Chapter 3, a possibility to improve the general robustness against Counter-Forensics, without specializing the forensic algorithm against a particular tool (as it is often the case with adversary-aware approaches,), is to resort to randomization strategies, in the attempt to design *generally more secure* detectors (see Section 3.3.2). In fact, several randomization approaches have been considered addressing standard machine learning tools and, more recently, DL architectures. In [16], the authors propose to randomize the selection of the feature space according to a secret key to prevent the attacker from gaining full knowledge about the system. In this way, the analyst *exits* the Perfect Knowledge (PK) scenario, or, by following the DL terminology [120], the *white-box* scenario, thus decreasing the success rate of the attack. Said differently, random feature selection induces a Limited Knowledge (LK) scenario for the attack.

The effectiveness of Random Feature Selection (RFS) has been proven in [16] both theoretically, under simplifying assumptions, and in practice, where it is experimentally validated by focusing on image manipulation detection and SVM-based classification. With regard to DL literature, most of the methods proposed so far focus on *test time randomization* [188], where the input layer of the classifier is randomized at test time. A multi-channel architecture, where each channel introduces its own randomization in a special transformed domain based on a secret key, has recently been proposed in [189].

Our goal is to extend the random feature selection approach described in [16] to the case of CNN-based detection, where the features are extracted by a convolutional neural network, to see if and to which extent the approach can be used to combat adversarial examples. This scheme is referred to as Random Deep Feature Selection (RDFS). To perform the classification based on the randomly selected deep features, we consider two types of classifiers, a Fully Connected (FC) network and a linear SVM. With regard to the FC network, it corresponds to a re-trained version of the classification part (the FC layers) of the original CNN targeted by the attack.

To be effective, RDFS should improve the security of CNN-based detectors against adversarial examples, at the expense of a negligible loss of performance in the absence of attacks.

It is easy to argue that the effectiveness of the RDFS scheme is directly related to the degree of transferability of the attack, hence to the amount of mismatch (in the detection architecture) that the attack is capable to withstand, while remaining effective.

## 8.2 Random Deep Feature Selection (RDFS) for Secure Image Classification

We now describe how we extended the random feature selection method developed in [16] for model-based and standard ML-based detectors based on statistical and handcrafted features, to the case of CNN-based forensic detectors. To be specific, the security model considered in this work is depicted in Figure 8.1: given an original CNN detector, the CNN is *only* used as feature extractor. Let $N$ be the dimensionality of the set of features. Then, $K$ features ($K < N$) are randomly selected among the $N$ features, according to a secret key. The reduced set of features obtained in this way is used to train another detector, for instance, a Neural Network or an SVM classifier. Obviously, the same scheme is applied during both training and testing, with the same secret key. Without loss of generality, we let $H_0$ be the hypothesis that the image is original, and $H_1$ the hypothesis that the image has been tampered with. The adversarial attack is carried out in the pixel domain, as shown in Figure 8.1. We assume that the attacker does not know the existence of the randomization strategy and then he targets the original CNN classifier (hence implementing a so called *vanilla attack*) [188, 190]. Also, we assume that the attacker wants to pass off a manipulated image as an original one, i.e., to induce the network to decide in favor of $H_0$ when $H_1$ holds, causing a false negative error. At the same time, the attacker wants to minimize the distortion introduced in the image as a consequence of the attack.

As mentioned in the previous section, the random feature selection en-

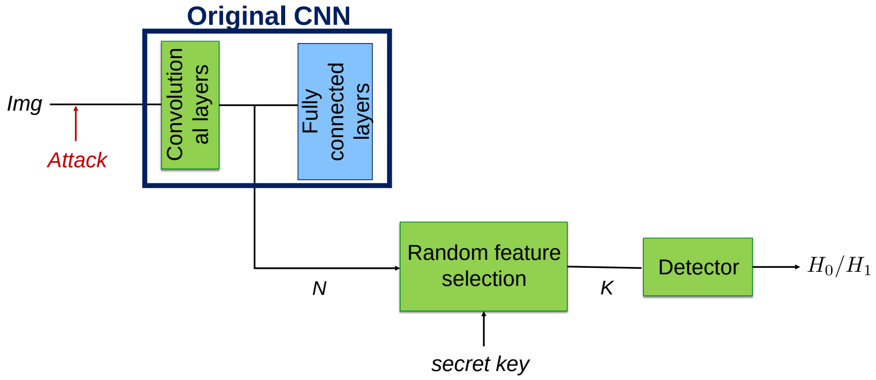


Figure 8.1: Scheme of the proposed RDFS detector.

forces a limited knowledge scenario for the attack. In particular, the attacker is not aware of the randomization defence mechanism, and, even if he is, he does not know the subset of features used by the detector, and their number $K$. Moreover, he has only a partial knowledge of the RDFS architecture of the detector. The exact amount of knowledge available to the attacker depends on the specific RDFS scheme considered, that is, on the specific architecture of the detector. Specifically, we considered two different scenarios: the case of a Fully-Connected (FC) network detector, and the case of a linear SVM detector.

### 8.2.1 RDFS detection based on a Fully-Connected (FC) network

In this case, the RDFS detector is implemented by retraining the FC layers of the original CNN. Given the selected random feature set of dimensionality $K$ ($K < N$), the same FC structure of the original network is re-trained considering only the $K$ input nodes. The number of layers depends then on the original CNN architecture. Therefore, when the full feature set is considered for the detection ($K = N$), there is no mismatch in the architecture between this FC network and the classification architecture of the original CNN targeted by the attack. However, even in this case, the attack is not completely white-box, since, in the setup considered for our experiments, a different training set (more precisely, a subset of the original set) is used to train the detector.

### 8.2.2 RDFS detection based on SVM

In this case, an SVM architecture is considered to implement the detector. Then, in contrast to the previous case, the detector adopts a different classification structure of the original CNN, even when the full feature set is considered ($K = N$). The amount of attack knowledge is less than in the previous case, since here the attacker does not know the architecture of the detector, in addition to the training data.

## 8.3 Application to Image Manipulation Detection

We are interested in evaluating the performance of the general RDFS approach introduced before for image manipulation detection applications, and investigate which are the factors that mostly affect its effectiveness. Specifically, we run several experiments by considering three different manipulation detection tasks. For each of them, two different state-of-the-art CNN architectures were considered. The security of the RDFS approach was assessed by considering three different types of attacks.

### 8.3.1 Original CNN detectors

We considered three different detection tasks, namely the detection of image resizing (downsampling, by a 0.8 factor), median filtering (by a 5 × 5 window), and adaptive histogram equalization (AHE), which applies contrast enhancement on a local basis (the Contrast-Limited implementation of AHE, CLAHE, was considered). Concerning the network architectures, we considered the same two networks considered in the previous chapter. In particular, they are the network originally proposed

### 8.3.2 Reduced (deep feature) detectors

For each manipulation task, we built our reduced feature detectors as detailed in the following. For a given choice of the random selection, the $K$ features were extracted from the flatten layers of the CNNs. Then, we trained the original FC architecture of BSnet and GCnet with $K$ input nodes, and a linear SVM classifier fed with the $K$ input features. Both the FC networks and the

SVM were trained on a subset of images of the training set used to train the original networks. Regarding the structure of the two FC networks, we have 2 layers with 4096 hidden nodes each for BSnet, and only one layer consisting of 250 hidden nodes for GCnet. For the SVM, a Gaussian kernel was adopted, and the kernel parameters were determined by 5-fold cross-validation.

To measure the performance of the reduced set detectors, for a given size $K$ of the reduced feature set, the experiments were repeated (both training and testing) 50 times, each time with a different randomly chosen feature subset.

### 8.3.3 Adversarial attacks

In all the cases, the attack is run against the original CNN model in a *white-box* setting, i.e., it is assumed that the attacker knows everything about the original model (note that assuming that the attacker attacks the original CNN we are implicitly assuming that the attacker is not aware of - and/or he does not react to - the defence mechanism).

In the experiments, we considered three gradient-based, iterative, attacks. Specifically, the adversarial examples were built by relying on the following algorithms: the original box constrained L-BFGS by Szegedy et al., the Iterative Fast Gradient Sign Method, namely I-FGSM, and the Projected Gradient Descent (PGD) attack (very similar to I-FGSM). We refer to Section 7.2 for an introduction on these attack algorithms. As for the experiments of the previous chapter, the Foolbox toolbox [187] was used to implement the attacks.

In order to increase the diversity of the attacks considered, in these experiments we also considered the L-BFGS attack: by looking for the 'minimum' adversarial perturbation, L-BFGS tends to generate adversarial examples with a very large PSNR, which are however (expectedly) less transferable compared to those obtained with most of the other (suboptimum) attack methods, like for instance FGSM.

## 8.4 Experimental Results

In this section, we first describe the experimental setup for our experiments, then we present and discuss the results of the tests we carried out in the various settings.

### 8.4.1 Experimental setup

For our experiments, we considered uncompressed camera-native (.tiff) images from RAISE-8K dataset. The images were split into training, validation, and test sets and then processed to create the images for the $H_1$ class, i.e., resizing, median, and adaptive histogram equalization (CLAHE). For all our experiments, the images were converted to gray-scale.

To build the original CNNs for the three detection tasks, we considered 100.000 patches for training, 3000 and 10.000 for validation and testing respectively, per class, for BSnet; for GCnet, 500.000 patches per class were considered for training, 5000 and 10.000 for validation and testing respectively. In order to use many images from the dataset and then enforce patch diversity, a maximum number of 100 patches were selected randomly from each image. For both networks, the input patch size was set to $64 \times 64$[1]. A number of 40 epochs was considered to train the BSnet models, and 4 epochs were considered for GCnet. For training the networks, we used the Adam solver with learning rate $10^{-4}$ and momentum 0.99. The batch size for training was set to 32. The accuracies achieved by the trained models in the absence of attacks are: i) BSnet: 91.30% for resizing, 98.83% for median filtering, and 90.45% for CLAHE; i) GCnet: 95.05% for resizing, 99.73% for median filtering, and 98.30% for CLAHE.

In order to build the RDFS detectors, both the FC networks and the SVM were trained on a subset of 20.000 patches per class selected from the original training set of images, and validated on 1000 patches per class, selected from

[1] A smaller input size corresponds to a smaller dimensionality of the feature set used for the classification (size of the flatten layer): specifically, for a $64 \times 64$ input size, the features are in the order of thousands (for both BSnet and GCnet), which is a good compromise (if, on one hand, the randomization is not effective if the feature set is too small, on the other hand, a too large dimensionality of the feature set complicates the analysis of the reduced deep feature detectors, requiring a lot of training)

the validation set used for the original CNN. Regarding the training procedure for the reduced feature FC networks, we used the following setting: learning rate of the Adam solver set to $10^{-5}$ and the momentum to 0.99 for a maximum number of 50 epochs, with an early stop condition that ends training when the validation loss changes less than $10^{-3}$ in the last 5 epochs (with a validation batch size of 100). The number of reduced features $K$ considered in our experiments (for training and testing) for both the FC networks and the SVM is: 5, 10, 30, 50, 200, 400, 600, and the full feature case $K = N$. The full feature set size (size of the flatten layer of the original CNN) is $N = 1728$ with BSnet and $N = 3200$ with GCnet. When GCnet is used as original CNN, the case $K = 600$ is not considered, to save time. For every value of $K$, in fact, we need to train 50 models (SVM and FC), one for every choice of the random set takes time. However, as confirmed by the results with BSnet, the most interesting cases are those with lower values of $K$ (order of tens). For the tests in the absence of attacks, we considered 4000 patches per class, taken from the original test set.

With regard to the attacks, for L-BFGS, we used the default attack parameters [187]. For I-FGSM, the number of steps $S$ was set to 10 (default), the best strength is searched in the range $E = [0 : 0.001, 0.1]$. PGD is applied considering the following setting: $\varepsilon = 0.05$, and $\alpha = 0.3$, binary-search = 'True' (binary-search = 'True' means that the input values of $\varepsilon$ and $\alpha$ are only used for the initialization, and the optimum choice of $\varepsilon$ and $\alpha$ is optimized via the binary search). The above setting does not work for the CLAHE detection task (the adversarial image cannot be found in most of the cases); for that task, the following setting has been considered: $\varepsilon = 0.025$, and $\alpha = 0.01$, binary-search = 'False'. As we said, we only applied the attack to images of the $H_1$ class. In all the cases, the performance in the presence of attacks is evaluated on 500 adversarial examples, obtained by attacking a subset of the 4000 patches of the $H_1$ class. We checked that, with the above setting, all the attacks are successful against the target original CNN, the success rate being in the range [0.98:1]. The average PSNR for the attacked samples is between 40 and 70dB (often above 60 dB), the exact value depending on the attack type, the target network and the detection task.

### 8.4.2 Results of FC-based classification

The results we have got for this case are reported in Table 8.1 and 8.2 for the case of BSnet and GCnet, respectively.[2]

Table 8.1: Accuracy (%) of the RDFS detector based on FC network, for the case of BSnet [8].

|  | Resize |  |  |  | Median Filtering |  |  |  | CLAHE |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| K | No Attk | **PGD** | **FGSM** | **BFGS** | No Attk | **PGD** | **FGSM** | **BFGS** | No Attk | **PGD** | **FGSM** | **BFGS** |
| 5 | 91.0 | 69.9 | 61.6 | 65.6 | 88.7 | 79.8 | 51.0 | 73.0 | 73.0 | 87.4 | 89.2 | 88.0 |
| 10 | 95.0 | 68.0 | 55.7 | 62.0 | 93.2 | 80.6 | 44.5 | 67.1 | 78.0 | 88.0 | 89.1 | 78.6 |
| 30 | 97.0 | 58.5 | 43.4 | 48.8 | 96.8 | 79.7 | 30.8 | 56.1 | 80.1 | 89.5 | 90.7 | 64.7 |
| 50 | 97.4 | 52.0 | 35.9 | 40.1 | 97.7 | 80.0 | 24.6 | 53.5 | 80.7 | 90.2 | 91.3 | 56.3 |
| 200 | 97.8 | 31.0 | 13.7 | 17.4 | 98.7 | 77.6 | 10.8 | 44.8 | 81.5 | 91.6 | 94.0 | 42.8 |
| 400 | 97.7 | 20.7 | 7.2 | 9.1 | 98.8 | 76.6 | 7.5 | 42.6 | 81.3 | 91.8 | 94.5 | 41.0 |
| 600 | 97.9 | 16.4 | 5.4 | 7.1 | 98.9 | 80.5 | 6.0 | 43.0 | 80.5 | 91.5 | 93.8 | 36.6 |
| $N$ | 98.0 | 31.5 | 0.6 | 20.3 | 99.0 | 81.9 | 4.3 | 39.7 | 80.5 | 91.8 | 93.9 | 35.1 |

By inspecting Table 8.1 we observe that, when the attack works well against the FC network with $K = N$ (last line of the tables), that is, when the attack targeted to the original BSnet model can be successfully transferred to the full feature FC detector, the proposed randomization strategy helps and a significant gain in the accuracy can be achieved, at the expense of a minor accuracy reduction in the absence of attacks. Specifically, the accuracy gain is about 20-30% for $K = 30$ and 30-50% for $K = 10$, while the accuracy reduction in the absence of attacks is only 2-4%, the exact value depending on the task (with the exception of the resizing detection task with GCnet, where a more significant loss of performance is experienced without attacks,

[2] L-BFGS and I-FGSM are referred to as BFGS and FGSM for short in the tables.

see Table 8.2).

Table 8.2: Accuracy (%) of the RDFS detector based on FC network, for the case of GCnet.

| | Resize | | | | Median Filtering | | | | CLAHE | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| K | No Attk | PGD | FGSM | BFGS | No Attk | PGD | FGSM | BFGS | No Attk | PGD | FGSM | BFGS |
| 5 | 74.4 | 67.7 | 58.7 | 60.7 | 97.2 | 83.3 | 48.3 | 77.1 | 87.4 | 47.2 | 63.7 | 47.0 |
| 10 | 78.6 | 71.9 | 59.9 | 63.0 | 98.8 | 86.1 | 44.3 | 79.2 | 91.1 | 55.6 | 68.8 | 48.3 |
| 30 | 92.7 | 81.8 | 65.5 | 70.7 | 99.4 | 88.5 | 30.0 | 79.6 | 94.3 | 56.7 | 76.3 | 39.8 |
| 50 | 96.8 | 85.2 | 66.8 | 73.0 | 99.6 | 87.4 | 21.9 | 76.6 | 95.1 | 50.6 | 80.0 | 35.3 |
| 200 | 99.7 | 88.0 | 69.6 | 77.9 | 99.6 | 88.6 | 17.0 | 76.2 | 96.9 | 48.5 | 83.0 | 26.0 |
| 400 | 99.8 | 89.3 | 71.8 | 80.0 | 99.6 | 88.1 | 15.6 | 75.6 | 97.1 | 30.1 | 83.6 | 21.0 |
| $N$ | 100 | 89.8 | 75.2 | 81.2 | 99.7 | 85.2 | 13.7 | 71.3 | 98.2 | 33.5 | 34.0 | 26.2 |

In some cases, however, it happens that the accuracy is already large also for $K = N$, i.e., the attack fails against the full feature FC detector, meaning that the attack targeted to the original CNN BSnet cannot be transferred to the full feature FC detector. Stated in another way, just re-training the FC network on a different set (a subset of the original training in our case) decreases by itself the attack success rate. This behavior confirms the findings of the previous chapter, showing that, at least for image forensic applications, the adversarial examples are generally non-transferable, in contrast to what happens in typical pattern recognition applications [13].

A similar behavior can be observed in Table 8.2, where we see that for $K = N$ the attack is even less effective than before (hence it is less transferable). Then, again, in these cases, the randomization defence is not necessary (e.g. for the case of PGD, the accuracy with $K = N$ is 89.7% for resizing and 85.3% for median filtering). In the other cases, i.e., when the accuracy with

$K = N$ is low, the randomization approach increases the accuracy by 20-30%.

### 8.4.3 Results of SVM-based classification

The results we have got for this case are reported in Table 8.3 and 8.4 for the case of BSnet and GCnet, respectively.

Table 8.3: Accuracy (%) of the RDFS detector based on SVM, for the case of BSnet [8].

| | Resize | | | | Median Filtering | | | | CLAHE | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| K | No Attk | PGD | FGSM | BFGS | No Attk | PGD | FGSM | BFGS | No Attk | PGD | FGSM | BFGS |
| 5 | 79.6 | 59.0 | 58.0 | 58.7 | 80.3 | 69.8 | 47.5 | 66.1 | 74.4 | 90.7 | 89.2 | 87.3 |
| 10 | 87.0 | 60.5 | 58.9 | 59.9 | 87.6 | 70.8 | 33.8 | 63.2 | 80.4 | 90.7 | 90.4 | 81.5 |
| 30 | 92.8 | 70.9 | 70.1 | 69.6 | 94.5 | 63.3 | 19.1 | 50.7 | 80.5 | 89.6 | 90.9 | 70.5 |
| 50 | 94.3 | 75.5 | 75.6 | 75.0 | 96.2 | 66.8 | 13.1 | 42.0 | 80.7 | 89.8 | 91.0 | 62.3 |
| 200 | 95.5 | 65.0 | 63.9 | 64.2 | 97.7 | 57.2 | 3.8 | 22.1 | 80.0 | 91.0 | 93.4 | 43.7 |
| 400 | 94.8 | 43.4 | 66.4 | 28.1 | 98.0 | 50.3 | 1.9 | 14.9 | 79.7 | 91.2 | 93.8 | 39.7 |
| 600 | 95.4 | 47.9 | 25.2 | 32.3 | 98.1 | 45.0 | 1.3 | 11.0 | 79.4 | 91.3 | 94.2 | 40.1 |
| $N$ | 95.1 | 58.4 | 31.0 | 39.4 | 98.0 | 29.6 | 0.6 | 5.0 | 79.5 | 91.8 | 95.0 | 38.6 |

In this case, expectedly, the mismatch in the architecture decreases further the success rate of the attack against the full feature SVM detector (case with $K = N$), i.e. it increases the accuracy, without even resorting to randomization. However, when this is not the case, randomization helps:

for instance, for BSnet under the L-BFGS attack, the accuracy passes from 39.4 (for the resizing task), 5.0 (for the median filtering task) and 38.6 (for the CLAHE task), to 69.6%, 50.7%, and 70.5 % respectively, with a per-

Table 8.4: Accuracy (%) of the RDFS detector based on SVM, for the case of GCnet.

| | Resize | | | | Median Filtering | | | | CLAHE | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| K | No Attk | **PGD** | **FGSM** | **BFGS** | No Attk | **PGD** | **FGSM** | **BFGS** | No Attk | **PGD** | **FGSM** | **BFGS** |
| 5 | 74.8 | 73.6 | 65.4 | 66.4 | 97.2 | 83.1 | 46.3 | 78.1 | 88.3 | 59.6 | 64.8 | 50.7 |
| 10 | 82.7 | 78.1 | 68.0 | 69.2 | 98.3 | 85.7 | 42.0 | 80.3 | 91.2 | 67.0 | 74.5 | 58.0 |
| 30 | 95.1 | 86.1 | 72.8 | 76.0 | 99.3 | 86.1 | 25.8 | 77.2 | 93.4 | 63.4 | 86.5 | 54.8 |
| 50 | 97.5 | 88.2 | 73.2 | 77.2 | 99.3 | 84.1 | 18.7 | 74.0 | 94.5 | 56.7 | 90.3 | 46.7 |
| 200 | 99.6 | 88.5 | 68.1 | 75.7 | 99.6 | 88.2 | 15.9 | 75.2 | 96.4 | 36.4 | 94.0 | 24.7 |
| 400 | 99.7 | 90.0 | 67.6 | 77.7 | 99.6 | 86.6 | 13.5 | 70.1 | 97.0 | 26.1 | 94.1 | 16.1 |
| $N$ | 99.8 | 90.6 | 66.2 | 83.8 | 99.7 | 86.4 | 12.0 | 69.8 | 97.3 | 22.3 | 94.6 | 11.0 |

formance loss without attacks of 2.2% in the accuracy. Using less features, e.g $K = 10$, the accuracy against the attacks can be improved further, though at the expense of a higher loss of performance without attacks.

## 8.5 Discussion and Remarks

Inspired by the work in [16], we have evaluated the feasibility of using deep feature randomization to improve the robustness of CNN detectors against adversarial examples. Our experiments carried out in a wide variety of scenarios reveal that feature randomization somewhat helps in decreasing the transferability of the attacks, hence improving the security of the detection. However, the degree of effectiveness of the RDFS scheme depends on the detection task, the kind of attack and the network, and, in some cases, the mismatch in the architecture, between the original CNN targeted by the attack and the one used for the classification (an SVM or the retrained FC layer) decreases by itself the attack success rate, that is, it is already enough to prevent the transferability of the attack, thus making the randomization unnecessary. Therefore, to better assess the effectiveness and understand the limits of the RDFS scheme, our current research is focused on increasing the strength of the attacks to improve the transferability of the adversarial examples, that is, to reinforce the attack (see the discussion at the end of Chapter 7).

As further research, it would be also interesting to investigate a scenario more favourable to the attacker, where the attacker is aware of the randomization-based defence. In particular, we can assume that the attacker is aware of the feature selection mechanism and the architecture of the detector (only the secret key is unknown), and then can run a more powerful attack, for instance by targeting an expected version of the classifier (in a way that resembles the Expectation over Transformation (EOT) attack [190]). From the defender's side, the use of a different FC layer or an SVM with a different kernel could be considered for the RDFS classification. More in general, one could try to improve the effectiveness of the RDFS scheme by performing feature regularization during the training of the original CNN, in such a way to reduce the gap with the theoretical analysis carried out in [16] about the effectiveness of the feature selection strategy. Finally, it would be also interesting to consider the application of the RDFS scheme in a black-box attack scenario, and assess the information leakage on the secret key in this case.

## Chapter 9

# Conclusion

*"When you reach the end of what you should know, you will be at the beginning of what you should sense."*
Kahlil Gibrán

*"You only grow by coming to the end of something and by beginning something else."*
John Irving

In this thesis, we proposed and investigated several different approaches for the systematic development of Machine Learning (ML)-based techniques for image manipulation detection capable to work in adversarial setting (Adversarial Multimedia Forensics), with specific reference to binary manipulation classification. In this chapter, we summarise the main contributions of the thesis, discuss important open issues, and point out possible directions for future research.

## 9.1 Summary

The first approach considered in this thesis for improving the robustness of machine learning tools against attacks was adversarial (or adversary-aware) training. We first introduced the concept of Most Powerful Attack (MPA) and proposed a MPA-aware detector, that can improve the security of ML-based image manipulation detection against a certain class of attacks. When the MPA cannot be find analytically, as it is often the case in practice, a possibility is to try to determine an approximation of the MPA experimentally, by looking for the attacks that cause most damage to the unaware system. The effectiveness of the MPA-aware training strategy has been validated by focusing on the problem of double JPEG detection, which is one of the most common and studied problems in Image Forensics.

Aware training is also considered as a way to improve the robustness of the forensic tools to JPEG post-processing. Several forensic tools, in fact, have been shown to have poor robustness to JPEG, as highlighted by recent research in the field. As a consequence, JPEG compression can also be considered and regarded to as a laundering-type Counter Forensic attack. Compared to more complicated attacks tailored for the various cases, as an advantage, the JPEG laundering attack is simple and easy to implement and does not require any knowledge of the target detector. With reference to the problem of contrast enhancement (CE) detection (which is well known to be a very challenging tasks in presence of JPEG post-processing), the effectiveness of JPEG-aware training has been assessed, for both standard ML (SVM-based detection) and DL (CNN-based detection).

The second approach considered in this thesis for improving the security of ML-based image manipulation detectors is the development of intrinsically more secure algorithms, that are obtained by working both on the architecture of the classifier and on the randomization of the feature set.

In the first case, we resorted to multiple classification. The one-and-a-half-class architecture that we evaluated, consisting of parallel of a 2C and two 1C classifiers followed by a final 1C classifier, is borrowed from the general literature of machine learning, and is purposely designed in the attempt to simultaneously get the advantages of 2C and 1C classification, that is, to yield at the same time good detection performance (similar to those obtained via 2C classifiers) and similar robustness to attacks as 1C classifiers.

We put the one-and-a-half-class approach at work by considering several different image manipulation detection tasks, namely, the detection of geometric operations, filtering and contrast enhancement. When applied to all these cases, the multiple classification architecture was proved to get very good results in terms of both robustness and security.

Recent studies in deep learning have shown that adversarial examples present a certain degree of transferability, implying that the attack can be effective even when it is carried out in a Limited Knowledge scenario. An extensive analysis of the transferability of the attacks that we carried out revealed that, when (state-of-the-art) Image Forensic CNN-based detectors are considered, adversarial examples are often non-transferable, in contrast to what happens in typical pattern recognition applications, where the transferability of the attack is much stronger. The general lack of transferability can then be exploited by the forensic analyst to make the attack more difficult, e.g. enforcing an LK scenario, as done with the approaches based on standard ML techniques. Feature randomization selection goes in this direction. In the case of CNNs, the approach consists in selecting a random set of features among those extracted from the convolutional part of the network. The results we got for three image manipulation detection tasks (resizing, median filtering and contrast enhancement), two network architectures and three classes of attacks, show that feature randomization indeed helps to mitigate the dangerousness of adversarial examples.

## 9.2 Open Issues

With this thesis, we contributed to the development of Machine Learning (ML)-based techniques for adversarial image manipulation detection. Several direction for future research can be pointed out, starting from this work.

In particular, the robustification of DL-based methods to adversarial examples is a very important direction for future research. From a more focused perspective, the effectiveness of deep feature randomization (RDFS) to improve the robustness of CNN detectors should be further tested against stronger attacks and also considering scenarios more favourable to the attacker, where the attacker is aware of the randomization-based defence. With regard to considering stronger attacks, as pointed out in the conclusions of Chapter 7, increasing the strength of the attack in such a way to obtain a more powerful attack (entering more inside the decision region of the target class) and then increase the attack transferability turns out to be a difficult task and then worth investigating by itself. Related to this, the fact that the adversarial examples against CNNs developed for image forensic tasks are generally less transferable is a remarkable result of this thesis, which deserves further investigation, being the starting point for the development of proper defence mechanisms. In order to test the effectiveness of the RDFS scheme in a more challenging adversarial scenario, more advantage should be given to the attacker by assuming that he is aware of the feature selection mechanism and the architecture of the detector (the only unknown being the secret key), and then run a more powerful *gray-box* attack targeting the RDFS classifier, for instance by attacking an expected version of the classifier.

The possibility of performing feature regularization during training could be investigated from the defender's side, to improve the effectiveness of deep feature randomization.

From a more general perspective, other more sophisticated randomization strategies could be considered to improve the security of DL against adversarial examples, where the gain in the security is provided by the secrecy of the randomization key.

A security threat in DL that will deserve more and more attention in the upcoming future is the one posed by the use of Generative Adversarial Networks (GANs) [125]. GANs have been used by the computer vision community to produce realistic images, thus posing themselves a forensic challenge, and there are already some works in the forensic literature investigating approaches for distinguishing GAN-generated images from real images. In this respect, the use of color rich feature sets (e.g., the CRM model), capable to expose possible (statistical) inconsistencies among the color bands of the images generated by GANs, or the analysis of the color co-occurrences, could be investigated. On the adversarial side, an attacker might use GANs to train a generator capable of falsifying forensic traces, and in fact, GAN-based Counter-Forensic attacks have already been proposed [126].

Finally, a new class of attacks against DL architectures which is worth mentioning is the class of so called backdoor attacks [191]. Backdoor attacks are about poisoning the training set (e.g. by injecting a backdoor signal to a portion of the images in the training set) so to ease inducing a classification error at test time, thus creating a 'backdoor' into the system. Although the link with image forensics is weak, backdoor attacks represent a serious threat in many security-related applications and then studying solutions to improve the security of DL algorithms against such attacks is of paramount importance.

The development of a new class of ML-based forensic tools thought to overcome the limitations of ML, and DL in particular, and to improve the robustness against attacks, is just in its fancy. Better understanding the capacities and limits of ML-based and DL-based attacks, and improving forensic techniques to protect or recognize these attacks as they emerge, will likely represent an important challenge for the years to come.

# Bibliography

[1] "Celebrities before-after photoshop." https://www.boredpanda.com/before-after-photoshop-celebrities/.

[2] "Photo tampering throughout history." http://pth.izitru.com/.

[3] R. Prabhu. Understanding of convolutional neural network (CNN)—deep learning. [Online]. Available: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-CNN-deep-learning-99760835f148

[4] Convolutional neural networks for visual recognition. [Online]. Available: http://cs231n.github.io/understanding-cnn/

[5] Beginners guide to convolutional neural networks. [Online]. Available: https://towardsdatascience.com/beginners-guide-to-understanding-convolutional-neural-networks-ae9ed58bb17d

[6] M. C. Stamm, S. K. Tjoa, W. S. Lin, and K. R. Liu, "Undetectable image tampering through JPEG compression anti-forensics," in *17th IEEE International Conference on Image Processing (ICIP)*, 2010, pp. 2109–2112.

[7] Z. Chen, B. Tondi, X. Li, R. Ni, Y. Zhao, and M. Barni, "A gradient-based pixel-domain attack against SVM detection of global image manipulations," in *2017 IEEE Workshop on Information Forensics and Security (WIFS)*, Dec 2017, pp. 1–6.

[8] B. Bayar and M. C. Stamm, "A deep learning approach to universal image manipulation detection using a new convolutional layer," in *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*, New York, NY, USA, 2016, pp. 5–10.

[9] "McAfee."

[10] R. Wortley and S. Smallbone, "Child pornography on the internet"." [Online]. Available: http://www.ncdsv.org/images/COPS_Child-Pornography-on-the-Internet_5-2006.pdf

[11] G. L. Wittel and S. F. Wu, "On attacking statistical spam filters," in *CEAS - First Conference on Email and Anti-Spam, Mountain View, California, USA*, 2004.

[12] D. Lowd and C. Meek, "Good word attacks on statistical spam filters," in *CEAS - Second Conference on Email and Anti-Spam, Stanford University, California, USA*, 2005.

[13] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv preprint arXiv:1605.07277*, 2016.

[14] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[15] A. Christmann and I. Steinwart, *Support Vector Machines*, Springer, ISBN 978-0-387-77242-4, 2008.

[16] Z. Chen, B. Tondi, X. Li, R. Ni, Y. Zhao, and M. Barni, "Secure detection of image manipulation by means of random feature selection," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 9, pp. 2454–2469, Sep. 2019.

[17] A. Costanzo, "Techniques for digital image forensics and counter-forensics," Ph.D. dissertation, University of Siena, 2014.

[18] M. Fontani, "Digital forensic techniques for splicing detection in multimedia contents," Ph.D. dissertation, University of Siena, 2015.

[19] M. Barni and F. Bartolini, *Watermarking Systems Engineering*, CRC Press, 2004.

[20] A. Piva, "An overview on image forensics," *ISRN Signal Processing*, vol. 2013, 01 2013.

[21] P. Ferrara, T. Bianchi, A. De Rosa, and A. Piva, "Image forgery localization via fine-grained analysis of CFA artifacts," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 5, pp. 1566–1577, Oct 2012.

[22] S. Bayram, H. Sencar, N. Memon, and I. Avcibas, "Source camera identification based on CFA interpolation," in *IEEE International Conference on Image Processing*, vol. 3, Sep. 2005, pp. III–69.

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

[24] N. Khanna, A. K. Mikkilineni, G. T.-C. Chiu, J. P. Allebach, and E. J. Delp, "Scanner identification using sensor pattern noise," in *Security, Steganography, and Watermarking of Multimedia Contents*, 2007.

[25] T. Gloe, E. Franz, and A. Winkler, "Forensics for flatbed scanners," in *Security, Steganography, and Watermarking of Multimedia Contents IX*, E. J. D. III and P. W. Wong, Eds., vol. 6505, International Society for Optics and Photonics. SPIE, 2007, pp. 541 – 552.

[26] H. Gou, A. Swaminathan, and M. Wu, "Robust scanner identification based on noise features - art. no. 65050s," *Proc SPIE*, vol. 6505, 2007.

[27] W. Luo, J. Huang, and G. Qiu, "JPEG error analysis and its applications to digital image forensics," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 3, pp. 480–491, Sep. 2010.

[28] T. Bianchi, A. De Rosa, and A. Piva, "Improved DCT coefficient analysis for forgery localization in JPEG images," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011, pp. 2444–2447.

[29] E. Nowroozi and A. Zakerolhosseini, "Double JPEG compression detection using statistical analysis," *Advances in Computer Science : an International Journal*, vol. 4, no. 3, pp. 70–76, 2015.

[30] H. Farid, "Digital image ballistics from JPEG quantization," Darmouth College, Department of Computer Science, Tech. Rep. TR2006-583, Hanoves, NH 2006.

[31] A. C. Popescu and H. Farid, "Statistical tools for digital forensics," in *Proceedings of the 6th International Conference on Information Hiding*, ser. IH'04. Berlin, Heidelberg: Springer-Verlag, 2004, pp. 128–147.

[32] F. Galvan, G. Puglisi, A. R. Bruna, and S. Battiato, "First quantization matrix estimation from double compressed JPEG images," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 8, pp. 1299–1310, Aug 2014.

[33] H. Farid, "Image forgery detection," *IEEE Signal Processing Magazine*, vol. 26, no. 2, pp. 16–25, March 2009.

[34] A. C. Popescu and H. Farid, "Exposing digital forgeries by detecting traces of resampling," *IEEE Transactions on Signal Processing*, vol. 53, no. 2, pp. 758–767, Feb 2005.

[35] G. Cao, Y. Zhao, and R. Ni, "Edge-based blur metric for tamper detection," *Journal of Information Hiding and Multimedia Signal Processing*, vol. 1, 2010.

[36] M. C. Stamm and K. R. Liu, "Forensic detection of image manipulation using statistical intrinsic fingerprints," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 3, pp. 492–506, 2010.

[37] M. Stamm and K. J. R. Liu, "Blind forensics of contrast enhancement in digital images," in *15th IEEE International Conference on Image Processing*, Oct 2008, pp. 3112–3115.

[38] M. Kirchner and J. Fridrich, "On detection of median filtering in digital images," in *SPIE Electronic Imaging*, vol. 7541, 2010, p. 754110.

[39] G. Cao, Y. Zhao, R. Ni, L. Yu, and H. Tian, "Forensic detection of median filtering in digital images," in *IEEE International Conference on Multimedia and Expo*, July 2010, pp. 89–94.

[40] H. Yuan, "Blind forensics of median filtering in digital images," *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 4, pp. 1335–1345, Dec 2011.

[41] B. Wang, X. Kong, and X. You, "Source camera identification using support vector machines," in *Advances in Digital Forensics V*, G. Peterson and S. Shenoi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 107–118.

[42] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*. ACM, 1992, pp. 144–152.

[43] Y. L. Cun and Y. Bengio, "The handbook of brain theory and neural networks." MIT Press, 1998.

[44] Y. Bengio, "Learning deep architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, Jan. 2009.

[45] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT'2010)*, Y. Lechevallier and G. Saporta, Eds. Paris, France: Springer, August 2010, pp. 177–187.

[46] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," *CoRR*, vol. abs/1206.5533, 2012.

[47] T. Pevny and J. Fridrich, “Detection of double-compression in JPEG images for applications in steganography,” *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 2, pp. 247–258, Jun. 2008.

[48] I. Amerini, R. Becarelli, R. Caldelli, and A. D. Mastio, “Splicing forgeries localization through the use of first digit features,” in *IEEE International Workshop on Information Forensics and Security (WIFS)*, Dec 2014, pp. 143–148.

[49] X. Feng and G. Doërr, “JPEG recompression detection,” in *Media Forensics and Security II*, N. D. Memon, J. Dittmann, A. M. Alattar, and E. J. D. III, Eds., vol. 7541, International Society for Optics and Photonics. SPIE, 2010, pp. 188 – 199.

[50] C. Chen, Y. Q. Shi, and W. Su, “A machine learning based scheme for double JPEG compression detection,” in *19th IEEE International Conference on Pattern Recognition*, 2008, pp. 1–4.

[51] Yi-Lei Chen and Chiou-Ting Hsu, “Image tampering detection by blocking periodicity analysis in JPEG compressed images,” in *IEEE 10th Workshop on Multimedia Signal Processing*, Oct 2008, pp. 803–808.

[52] S. Milani, M. Tagliasacchi, and S. Tubaro, “Discriminating multiple JPEG compression using first digit features,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2012, pp. 2253–2256.

[53] B. Li, Y. Q. Shi, and J. Huang, “Detecting doubly compressed JPEG images by using mode based first digit features,” in *IEEE 10th Workshop on Multimedia Signal Processing,*, 2008, pp. 730–735.

[54] N. Singh, A. Gupta, and R. Jain, “Global contrast enhancement based image forensics using statistical features,” *Advances in Electrical and Electronic Engineering*, vol. 15, 10 2017.

[55] M. M. Islam, G. Karmakar, J. Kamruzzaman, M. Murshed, G. Kahandawa, and N. Parvin, “Detecting splicing and copy-move attacks in color images,” in *Digital Image Computing: Techniques and Applications (DICTA)*, Dec 2018, pp. 1–7.

[56] M. Kharrazi, H. T. Sencar, and N. Memon, “Blind source camera identification,” in *International Conference on Image Processing, 2004. ICIP ’04.*, vol. 1, Oct 2004, pp. 709–712 Vol. 1.

[57] K. San Choi, E. Lam, and K. Wong, “Automatic source camera identification using the intrinsic lens radial distortion,” *Optics express*, vol. 14, pp. 11 551–65, 2006.

[58] T. Filler, J. Fridrich, and M. Goljan, “Using sensor pattern noise for camera model identification,” in *15th IEEE International Conference on Image Processing*, Oct 2008, pp. 1296–1299.

[59] J. Fridrich and J. Kodovsky, “Rich models for steganalysis of digital images,” *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 868–882, 2012.

[60] M. Goljan, J. Fridrich, and R. Cogranne, “Rich model for steganalysis of color images,” in *IEEE International Workshop on Information Forensics and Security (WIFS)*, 2014, pp. 185–190.

[61] C. Chen and M. C. Stamm, “Camera model identification framework using an ensemble of demosaicing features,” in *IEEE International Workshop on Information Forensics and Security (WIFS)*, Nov 2015, pp. 1–6.

[62] F. Marra, G. Poggi, C. Sansone, and L. Verdoliva, “Evaluation of residual-based local features for camera model identification,” in *New Trends in Image Analysis and Processing – ICIAP Workshops*, V. Murino, E. Puppo, D. Sona, M. Cristani, and C. Sansone, Eds. Springer International Publishing, 2015, pp. 11–18.

[63] A. Tuama, F. Comby, and M. Chaumont, “Camera model identification based machine learning approach with high order statistics features,” in *24th European Signal Processing Conference (EUSIPCO)*, Aug 2016, pp. 1183–1187.

[64] L. Verdoliva, D. Cozzolino, and G. Poggi, “A feature-based approach for image tampering detection and localization,” in *IEEE International Workshop on Information Forensics and Security (WIFS)*, 2014, pp. 149–154.

[65] G. Cao, Y. Zhao, R. Ni, and X. Li, “Contrast enhancement-based forensics in digital images,” *IEEE transactions on information forensics and security*, vol. 9, no. 3, pp. 515–525, 2014.

[66] H. Li, W. Luo, X. Qiu, and J. Huang, “Identification of various image operations using residual-based features,” *IEEE Transactions on Circuits and Systems for Video Technology*, 2016.

[67] Wei Wang, J. Dong, and T. Tan, “Effective image splicing detection based on image chroma,” in *16th IEEE International Conference on Image Processing (ICIP)*, Nov 2009, pp. 1257–1260.

[68] D. Cozzolino, D. Gragnaniello, and L. Verdoliva, “Image forgery detection through residual-based local descriptors and block-matching,” in *IEEE International Conference on Image Processing (ICIP)*, 2014, pp. 5297–5301.

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

[70] Y. Qian, J. Dong, W. Wang, and T. Tan, “Deep learning for steganalysis via convolutional neural networks,” in *Media Watermarking, Security, and Forensics*, A. M. Alattar, N. D. Memon, and C. D. Heitzenrater, Eds., vol. 9409, International Society for Optics and Photonics. SPIE, 2015, pp. 171 – 180.

[71] L. Pibre, J. Pasquet, D. Ienco, and M. Chaumont, “Deep learning for steganalysis is better than a rich model with an ensemble classifier, and is natively robust to the cover source-mismatch,” *ArXiv*, vol. abs/1511.04855, 2015.

[72] G. Xu, H. Wu, and Y. Shi, “Structural design of convolutional neural networks for steganalysis,” *IEEE Signal Processing Letters*, vol. 23, no. 5, pp. 708–712, May 2016.

[73] J. Chen, X. Kang, Y. Liu, and Z. J. Wang, “Median filtering forensics based on convolutional neural networks,” *IEEE Signal Processing Letters*, vol. 22, no. 11, pp. 1849–1853, Nov 2015.

[74] B. Bayar and M. Stamm, “Constrained convolutional neural networks: a new approach towards general purpose image manipulation detection,” *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 11, pp. 2691–2706, 2018.

[75] A. Tuama, F. Comby, and M. Chaumont, “Camera model identification with the use of deep convolutional neural networks,” in *IEEE International Workshop on Information Forensics and Security (WIFS)*, Dec 2016, pp. 1–6.

[76] L. Bondi, L. Baroffio, D. Guera, P. Bestagini, E. Delp, and S. Tubaro, “First steps toward camera identification with convolutional neural networks,” *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 259–263, 2017.

[77] Q. Wang and R. Zhang, “Double JPEG compression forensics based on a convolutional neural network,” *EURASIP Journal on Information Security*, vol. 2016, 12 2016.

[78] M. Barni, L. Bondi, N. Bonettini, P. Bestagini, A. Costanzo, M. Maggini, B. Tondi, and S. Tubaro, “Aligned and non-aligned double JPEG detection using convolutional neural networks,” *Journal of Visual Communication and Image Representation*, vol. 49, pp. 153–163, 2017.

[79] Y. Niu, B. Tondi, Y. Zhao, and M. Barni, “Primary quantization matrix estimation of double compressed JPEG images via CNN,” *IEEE Signal Processing Letters*, vol. 27, pp. 191–195, 2020.

[80] D. Cozzolino and L. Verdoliva, “Noiseprint: A CNN-based camera model fingerprint,” *IEEE Transactions on Information Forensics and Security*, vol. PP, pp. 1–1, 05 2019.

[81] P. Zhou, X. Han, V. I. Morariu, and L. S. Davis, “Learning rich features for image manipulation detection,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2018, pp. 1053–1061.

[82] Y. Chen, X. Kang, Y. Shi, and Z. Wang, “A multi-purpose image forensic method using densely connected convolutional neural networks,” *Journal of Real-Time Image Processing*, vol. 16, 2019.

[83] X. Huang, S. Wang, and G. Liu, “Detecting double JPEG compression with same quantization matrix based on dense CNN feature,” in *25th IEEE International Conference on Image Processing (ICIP)*, Oct 2018, pp. 3813–3817.

[84] D.-T. Dang-Nguyen, C. Pasquini, V. Conotter, and G. Boato, “RAISE: A raw images dataset for digital image forensics,” in *Proceedings of the 6th ACM Multimedia Systems Conference*, ser. MMSys ’15, New York, NY, USA, 2015, pp. 219–224.

[85] D. Shullani, M. Fontani, M. Iuliani, O. Shaya, and A. Piva, “VISION: a video and image dataset for source identification,” *EURASIP Journal on Information Security*, pp. 1–16, 2017.

[86] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, “Can machine learning be secure?” in *Proceedings of the ACM Symposium on Information, Computer and Communications Security*, ser. ASIACCS ’06, New York, NY, USA, 2006, pp. 16–25.

[87] M. Barni, M. C. Stamm, and B. Tondi, “Adversarial multimedia forensics: Overview and challenges ahead,” in *26th European Signal Processing Conference (EUSIPCO)*. IEEE, 2018, pp. 962–966.

[88] M. Kirchner and R. Böhme, “Tamper hiding: Defeating image forensics,” in *Information Hiding*, T. Furon, F. Cayre, G. Doërr, and P. Bas, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 326–341.

[89] M. Kirchner and R. Bohme, “Hiding traces of resampling in digital images,” *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 4, pp. 582–592, Dec 2008.

[90] G. Cao, Y. Zhao, R. Ni, and H. Tian, “Anti-forensics of contrast enhancement in digital images,” in *Proceedings of the 12th ACM Workshop on Multimedia and Security*, 2010, pp. 25–34.

[91] M. C. Stamm and K. J. R. Liu, "Anti-forensics of digital image compression," *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 3, pp. 1050–1065, Sep. 2011.

[92] G. Valenzise, V. Nobile, M. Tagliasacchi, and S. Tubaro, "Countering JPEG anti-forensics," in *18th IEEE International Conference on Image Processing*, Sep. 2011, pp. 1949–1952.

[93] H. Li, W. Luo, and J. Huang, "Countering anti-JPEG compression forensics," in *2012 19th IEEE International Conference on Image Processing*, Sep. 2012, pp. 241–244.

[94] G. Valenzise, M. Tagliasacchi, and S. Tubaro, "Revealing the traces of JPEG compression anti-forensics," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 2, pp. 335–349, Feb 2013.

[95] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. Tygar, "Adversarial machine learning," in *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, 2011, pp. 43–58.

[96] B. Biggio, G. Fumera, and F. Roli, "Security evaluation of pattern classifiers under attack," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 4, pp. 984–996, April 2014.

[97] B. Biggio, I. Corona, Z.-M. He, P. P. K. Chan, G. Giacinto, D. S. Yeung, and F. Roli, "One-and-a-half-class multiple classifier systems for secure learning against evasion attacks at test time," in *Multiple Classifier Systems*, F. Schwenker, F. Roli, and J. Kittler, Eds. Cham: Springer International Publishing, 2015, pp. 168–180.

[98] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Machine Learning and Knowledge Discovery in Databases*, H. Blockeel, K. Kersting, S. Nijssen, and F. Železný, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 387–402.

[99] M. Barni, M. Fontani, and B. Tondi, "A universal technique to hide traces of histogram-based image manipulations," in *Proceedings of the on Multimedia and Security*, ser. MMSec '12. New York, NY, USA: ACM, 2012, pp. 97–104.

[100] M. Barni, M. Fontani, and B. Tondi, "Universal counterforensics of multiple compressed JPEG images," in *International Workshop on Digital Watermarking*. Springer, 2014, pp. 31–46.

[101] R. Böhme and M. Kirchner, *Counter-Forensics: Attacking Image Forensics.* New York, NY: Springer New York, 2013, pp. 327–366.

[102] C. Pasquini, P. Comesaña-Alfaro, F. Pérez-González, and G. Boato, "Transportation-theoretic image counterforensics to first significant digit histogram forensics," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 2699–2703.

[103] P. Comesana and F. Perez-Gonzalez, "The optimal attack to histogram-based forensic detectors is simple(x)," in *IEEE International Workshop on Information Forensics and Security (WIFS)*, Dec 2014, pp. 137–142.

[104] M. Fontani and M. Barni, "Hiding traces of median filtering in digital images," in *Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, Aug 2012, pp. 1239–1243.

[105] I. Amerini, M. Barni, R. Caldelli, and A. Costanzo, "Counter-forensics of SIFT-based copy-move detection by means of keypoint classification," *EURASIP Journal on Image and Video Processing*, vol. 2013, p. 18, 03 2013.

[106] F. Marra, G. Poggi, F. Roli, C. Sansone, and L. Verdoliva, "Counter-forensics in machine learning based forgery detection," in *Media Watermarking, Security, and Forensics*, vol. 9409. International Society for Optics and Photonics, 2015, p. 94090L.

[107] C. Villani, *Optimal Transport: Old and New*, Springer, ISBN 978-3-540-71050-9, 2009.

[108] B. Tondi, "Theoretical foundations of adversarial detection and applications to multimedia forensics," Ph.D. dissertation, University of Siena, 2016.

[109] P. Comesaña-Alfaro and F. Pérez-González, "Optimal counterforensics for histogram-based forensics," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 3048–3052.

[110] T. Gloe, M. Kirchner, A. Winkler, and R. Böhme, "Can we trust digital image forensics?" in *ACM Multimedia*, 2007.

[111] F. Zhang, P. P. K. Chan, B. Biggio, D. S. Yeung, and F. Roli, "Adversarial feature selection against evasion attacks," *IEEE Transactions on Cybernetics*, vol. 46, no. 3, pp. 766–777, March 2016.

[112] J. Kodovskỳ, V. Sedighi, and J. Fridrich, "Study of cover source mismatch in steganalysis and ways to mitigate its impact," in *IS&T/SPIE Electronic Imaging.* International Society for Optics and Photonics, 2014, pp. 90 280J–90 280J.

[113] P. Sutthiwan and Y. Q. Shi, “Anti-forensics of double JPEG compression detection,” in *Digital Forensics and Watermarking*, Y. Q. Shi, H.-J. Kim, and F. Perez-Gonzalez, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 411–424.

[114] Z. Wu, M. C. Stamm, and K. J. R. Liu, “Anti-forensics of median filtering,” in *IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 3043–3047.

[115] O. Mayer and M. Stamm, “Anti-forensics of chromatic aberration,” *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 9409, 03 2015.

[116] F. Marra, D. Gragnaniello, and L. Verdoliva, “On the vulnerability of deep learning to adversarial attacks for camera model identification,” *Signal Processing: Image Communication*, vol. 65, pp. 240–248, July, 2018.

[117] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” *arXiv preprint arXiv:1312.6199*, 2013.

[118] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, “Deepfool: A simple and accurate method to fool deep neural networks,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 2574–2582.

[119] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, “The limitations of deep learning in adversarial settings,” in *IEEE European Symposium on Security and Privacy (EuroS P)*, March 2016, pp. 372–387.

[120] X. Yuan, P. He, Q. Zhu, and X. Li, “Adversarial examples: Attacks and defenses for deep learning,” *IEEE transactions on neural networks and learning systems*, 2019.

[121] Z. Zheng and P. Hong, “Robust detection of adversarial attacks by modeling the intrinsic properties of deep neural networks,” in *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, ser. NIPS’18. USA: Curran Associates Inc., 2018, pp. 7924–7933.

[122] D. Guera, Y. Wang, L. Bondi, P. Bestagini, S. Tubaro, and E. J. Delp, “A counter-forensic method for CNN-based camera model identification,” in *IEEE Computer Vision and Pattern Recognition Workshops*, July 2017, pp. 1840–1847.

[123] D. Gragnaniello, F. Marra, G. Poggi, and L. Verdoliva, “Analysis of adversarial attacks against CNN-based image forgery detectors,” *26th European Signal Processing Conference (EUSIPCO)*, pp. 967–971, 2018.

[124] B. Tondi, “Pixel-domain adversarial examples against CNN-based manipulation detectors,” *Electronics Letters*, 2018.

[125] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680.

[126] D. Kim, H.-U. Jang, S.-M. Mun, S. Choi, and H.-K. Lee, “Median filtered image restoration and anti-forensics using adversarial networks,” *IEEE Signal Processing Letters*, vol. PP, pp. 1–1, 2017.

[127] Y. Luo, H. Zi, Q. Zhang, and X. Kang, “Anti-forensics of jpeg compression using generative adversarial networks,” in *26th European Signal Processing Conference (EUSIPCO)*, Sep. 2018, pp. 952–956.

[128] S. Lai and R. Böhme, “Countering counter-forensics: The case of JPEG compression,” in *Information Hiding*, T. Filler, T. Pevný, S. Craver, and A. Ker, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 285–298.

[129] H. Zeng, T. Qin, X. Kang, and L. Liu, “Countering anti-forensics of median filtering,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 2704–2708.

[130] O. Mayer and M. C. Stamm, “Countering anti-forensics of lateral chromatic aberration,” in *Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security*, ser. IH&#38;MMSec ’17, New York, NY, USA, 2017, pp. 15–20.

[131] M. Goljan, J. Fridrich, and M. Chen, “Defending against fingerprint-copy attack in sensor-based camera identification,” *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 1, pp. 227–236, March 2011.

[132] A. Costanzo, I. Amerini, R. Caldelli, and M. Barni, “Forensic analysis of SIFT keypoint removal and injection,” *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 9, pp. 1450–1464, Sep. 2014.

[133] M. Barni, Z. Chen, and B. Tondi, “Adversary-aware, data-driven detection of double JPEG compression: How to make counter-forensics harder,” in *2016 IEEE International Workshop on Information Forensics and Security (WIFS)*, Dec 2016, pp. 1–6.

[134] M. Boroumand and J. Fridrich, “Scalable processing history detector for JPEG images,” *Electronic Imaging*, vol. 2017, pp. 128–137, 01 2017.

[135] M. Barni and F. Pérez-Gonzàlez, "Coping with the enemy: Advances in adversary-aware signal processing," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 8682–8686.

[136] 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.

[137] X. Lin, C. Li, and Y. Hu, "Exposing image forgery through the detection of contrast enhancement," in *IEEE International Conference on Image Processing*, Sep. 2013, pp. 4467–4471.

[138] A. De Rosa, M. Fontani, M. Massai, A. Piva, and M. Barni, "Second-order statistics analysis to cope with contrast enhancement counter-forensics," *IEEE Signal Processing Letters*, vol. 22, no. 8, pp. 1132–1136, Aug 2015.

[139] G. Singh and K. Singh, "Counter JPEG anti-forensic approach based on the second-order statistical analysis," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 5, pp. 1194–1209, May 2019.

[140] X. Pan, X. Zhang, and S. Lyu, "Exposing image forgery with blind noise estimation," in *Proceedings of the thirteenth ACM multimedia workshop on Multimedia and security*, 2011, pp. 15–20.

[141] M. Fontani, A. Bonchi, A. Piva, and M. Barni, "Countering anti-forensics by means of data fusion," in *Media Watermarking, Security, and Forensics*, 2014.

[142] M. J. Osborne and A. Rubinstein, *A Course in Game Theory*, MIT Press, 1994.

[143] R. B. Myerson, *Game Theory*, Harvard University Press, 1997.

[144] M. C. Stamm, W. S. Lin, and K. J. R. Liu, "Temporal forensics and anti-forensics for motion compensated video," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 4, pp. 1315–1329, Aug 2012.

[145] M. C. Stamm, W. S. Lin, and K. J. R. Liu, "Forensics vs. anti-forensics: A decision and game theoretic framework," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2012, pp. 1749–1752.

[146] M. Barni and B. Tondi, "The source identification game: An information-theoretic perspective," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 3, pp. 450–463, March 2013.

[147] M. Barni and B. Tondi, "Binary hypothesis testing game with training data," *IEEE Transactions on Information Theory*, vol. 60, no. 8, pp. 4848–4866, Aug 2014.

[148] M. Barni and B. Tondi, “Adversarial source identification game with corrupted training,” *IEEE Transactions on Information Theory*, vol. 64, no. 5, pp. 3894–3915, May 2018.

[149] M. Barni and B. Tondi, “Source distinguishability under distortion-limited attack: An optimal transport perspective,” *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 10, pp. 2145–2159, Oct 2016.

[150] S. Milani, M. Tagliasacchi, and S. Tubaro, “Antiforensics attacks to Benford’s law for the detection of double compressed images,” in *IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 3053–3057.

[151] J. Kodovskỳ and J. Fridrich, “Steganalysis of JPEG images using rich models,” in *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics, 2012, pp. 83 030A–83 030A.

[152] T. Pevny and J. Fridrich, “Merging Markov and DCT features for multi-class JPEG steganalysis,” *Proc. SPIE*, vol. 6505, pp. 650 503–650 503–13, 2007.

[153] G. K. Wallace, “The JPEG still picture compression standard,” *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pp. xviii–xxxiv, Feb 1992.

[154] M. K. Mihcak, I. Kozintsev, and K. Ramchandran, “Spatially adaptive statistical modeling of Wavelet image coefficients and its application to denoising,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 6, 1999, pp. 3253–3256.

[155] K. Zuiderveld, “Graphics gems iv,” P. S. Heckbert, Ed. San Diego, CA, USA: Academic Press Professional, Inc., 1994, ch. Contrast Limited Adaptive Histogram Equalization, pp. 474–485.

[156] N. Singh and A. Gupta, “Analysis of contrast enhancement forensics in compressed and uncompressed images,” in *IEEE Internationa Conference on Signal Processing and Communication (ICSC)*, 2016, pp. 303–307.

[157] G. Valenzise, M. Tagliasacchi, and S. Tubaro, “The cost of JPEG compression anti-forensics,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2011, pp. 1884–1887.

[158] T. Gloe and R. Böhme, “The ’Dresden Image Database’ for benchmarking digital image forensics,” in *Proceedings of the ACM Symposium on Applied Computing*, ser. SAC ’10, New York, NY, USA, 2010, pp. 1584–1590.

[159] G. Cao, Y. Zhao, and R. Ni, "Forensic estimation of gamma correction in digital images," in *17th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2010, pp. 2097–2100.

[160] S. M. Pizer, E. P. Amburn, J. D. Austin, R. Cromartie, A. Geselowitz, T. Greer, B. ter Haar Romeny, J. B. Zimmerman, and K. Zuiderveld, "Adaptive histogram equalization and its variations," *Computer vision, graphics, and image processing*, vol. 39, no. 3, pp. 355–368, 1987.

[161] P. Bestagini, A. Allam, S. Milani, M. Tagliasacchi, and S. Tubaro, "Video codec identification," in *IEEE Internation conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 2257–2260.

[162] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[163] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[164] F. Chollet *et al.*, "Keras," https://github.com/keras-team/keras, 2015.

[165] D. D'Avino, D. Cozzolino, G. Poggi, and L. Verdoliva, "Autoencoder with recurrent neural networks for video forgery detection," *Electronic Imaging*, vol. 2017, no. 7, pp. 92–99, 2017.

[166] E. Marchi, F. Vesperini, F. Eyben, S. Squartini, and B. Schuller, "A novel approach for automatic acoustic novelty detection using a denoising autoencoder with bidirectional LSTM neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2015, pp. 1996–2000.

[167] M. Hasan, J. Choi, J. Neumann, A. K. Roy-Chowdhury, and L. S. Davis, "Learning temporal regularity in video sequences," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 733–742.

[168] R. Perdisci, G. Gu, and W. Lee, "Using an ensemble of one-class SVM classifiers to harden payload-based anomaly detection systems," in *Sixth International Conference on Data Mining (ICDM'06)*, Dec 2006, pp. 488–498.

[169] S. S. Khan and M. G. Madden, "A survey of recent trends in one class classification," in *Artificial Intelligence and Cognitive Science*, L. Coyle and J. Freyne, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 188–197.

[170] A. Rattani, W. J. Scheirer, and A. Ross, "Open set fingerprint spoof detection across novel fabrication materials," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 11, pp. 2447–2460, Nov 2015.

[171] F. d. O. Costa, M. Eckmann, W. J. Scheirer, and A. Rocha, "Open set source camera attribution," in *2012 25th SIBGRAPI Conference on Graphics, Patterns and Images*, Aug 2012, pp. 71–78.

[172] B. Bayar and M. C. Stamm, "Towards open set camera model identification using a deep learning framework," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2018, pp. 2007–2011.

[173] B. Wang, X. Kong, and X. You, "Source camera identification using support vector machines," in *Advances in Digital Forensics V*, G. Peterson and S. Shenoi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 107–118.

[174] S. Yarlagadda, D. Güera, P. Bestagini, F. Zhu, S. Tubaro, and E. Delp, "Satellite image forgery detection and localization using gan and one-class classifier," *Electronic Imaging*, vol. 2018, 02 2018.

[175] P. Zheng, S. Yuan, X. Wu, J. Y. Li, and A. Lu, "One-class adversarial nets for fraud detection," in *The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*, 2018.

[176] J. Kodovsky, J. Fridrich, and V. Holub, "Ensemble classifiers for steganalysis of digital media," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 432–444, 2012.

[177] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Comput.*, vol. 13, no. 7, pp. 1443–1471, Jul. 2001.

[178] B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett, "New support vector algorithms," *Neural Comput.*, vol. 12, no. 5, pp. 1207–1245, May 2000.

[179] D. C. Montgomery, *Design and analysis of experiments.* John wiley & sons, 2017.

[180] C.-W. Hsu, C.-C. Chang, C.-J. Lin *et al.*, "A practical guide to support vector classification," 2003.

[181] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, April 2004.

[182] B. Tondi, "Pixel-domain adversarial examples against CNN-based manipulation detectors," *Electronics Letters*, vol. 54, no. 21, pp. 1220–1222, 2018.

[183] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," *arXiv preprint arXiv:1608.04644, 2016.*

[184] N. Akhtar and M. Ajmal, "Threat of adversarial attacks on deep learning in computer vision: a survey," *IEEE Access*, vol. 2018, no. 6, pp. 14 410–14 430, 2018.

[185] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations*, 2018.

[186] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[187] J. Rauber, W. Brendel, and M. Bethge, "Foolbox v0. 8.0: A python toolbox to benchmark the robustness of machine learning models," *arXiv preprint arXiv:1707.04131*, 2017.

[188] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. Yuille, "Mitigating adversarial effects through randomization," *arXiv preprint arXiv:1711.01991*, 2017.

[189] O. Taran, S. Rezaeifar, T. Holotyak, and S. Voloshynovskiy, "Defending against adversarial attacks by randomized diversification," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[190] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," *arXiv preprint arXiv:1802.00420*, 2018.

[191] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *Arxiv, CoRR*, vol. abs/1708.06733, 2017.

# Index

# Author's Information

Ehsan Nowroozi, the author of this thesis, was born on the 10th of June 1986 in Iran. He grew up in Shiraz city is one of the most beautiful, historical cities in the world. Farsi (Persian or Parsi) the language of Ancient Fars (Pars), has become the official language of Iran (Persia). The first Capital of Fars, some 2500 years ago, was Pasargad. It was also the capital of Achaemenid King Cyrus the Great. His doctoral research investigates in Multimedia Forensics, with particular reference to: *Machine learning techniques for image forensics in an adversarial setting.* He holds a master's degree in Computer Engineering - Computer Architecture from the Shahid Beheshti University (SBU), Tehran, Iran, that investigated *Double JPEG compression detection using statistical analysis.* He graduated as the first rank during his M.Sc with GPA 17.95 out of 20. In October 2016, he was selected as a Ph.D. student with the scholarship from the University of Siena, Italy in information engineering and mathematical sciences, working in the Visual Information Processing and Protection (VIPP) Lab under the supervision of Professor Mauro Barni. He wins one research scholarships, which supported by Defense Advanced Reseach Projects Agency (DARPA) and US Airforce Laboratory, USA. He has published various papers and a couple of books (Persian language) in the field of multimedia forensics. Also, he has been working on theoretical and practical aspects of adversarial multimedia forensics and adversarial machine learning with particular reference to the application of image processing techniques to authentication of multimedia (multimedia forensics).

📱 Scan me

The use of machine-learning for multimedia forensics is gaining more and more consensus, especially due to the amazing possibilities offered by modern machine learning techniques. By exploiting deep learning tools, new approaches have been proposed whose performance remarkably exceed those achieved by state-of-the-art methods based on standard machine-learning and model-based techniques. However, the inherent vulnerability and fragility of machine learning architectures pose new serious security threats, hindering the use of these tools in security-oriented applications, and, among them, multimedia forensics. The analysis of the security of machine learning-based techniques in the presence of an adversary attempting to impede the forensic analysis, and the development of new solutions capable to improve the security of such techniques is then of primary importance, and, recently, has marked the birth of a new discipline, named Adversarial Machine Learning.

By focusing on Image Forensics and image manipulation detection in particular, this thesis contributes to the above mission by developing novel techniques for enhancing the security of binary manipulation detectors based on machine learning in several adversarial scenarios. The validity of the proposed solutions has been assessed by considering several manipulation tasks, ranging from the detection of double compression and contrast adjustment, to the detection of geometric transformations and filtering operations.

UNIVERSITÀ
DI SIENA
1240

The Ph.D. School of Information Engineering of the University of Siena is a school aiming at educating scholars in a number of fields of research in the Information Engineering area. The Ph.D. School of Information Engineering is part of the Santa Chiara High School of the University of Siena. A Scientific Committee of external experts recognized Ph.D. Schools belonging to Santa Chiara as excellent, according to their degree of internationalization, their research, and educational activities.