POLITECNICO DI MILANO
Scuola di Ingegneria Industriale e dell'Informazione
Corso di Laurea Magistrale in Ingegneria Matematica

IMAGE TAMPERING DETECTION AND LOCALIZATION

Relatore: Prof. Stefano Tubaro

Tesi di Laurea di:
Lorenzo Gaborini, matricola 769924

Anno Accademico 2013-2014

Abstract

Image tampering is nowadays at everyone's reach: this has determined the urgent necessity of forensic tools capable of blindly distinguishing whether and where an image has been altered. To do so without having any prior information on the examined images reveals to be a challenging task.

The goal of this work is to propose two algorithms respectively aimed at classifying whether any given image is fake or not, and locating such tampering. These algorithms are inspired by those who won the IEEE First Image Forensic Challenge, the first competition organized specifically to compare state-of-the-art methods on blind image forensics. In particular, the first algorithm is strongly based on a machine learning approach, while the second one is a fusion between two entirely novel techniques with another well-known method.

Results, validated against the same dataset used in the Challenge, show that in both cases our proposed algorithms reach or beat the state-of-the-art performance on the specific dataset, yet leaving many questions open to discussion and further improvement. As a side effect of a deep joint analysis of the Challenge dataset along with the first proposed algorithm, we also uncovered a new forensic fingerprint which could be used for other attacks, proving also that it has been unknowingly exploited in the winning submission.

Estratto

Oggigiorno, il fotoritocco è alla portata di chiunque: da qui la necessità di strumenti forensi in grado di discriminare immagini autentiche da immagini ritoccate. Fare ciò in modo completamente alla cieca, senza informazioni a priori sulle immagini esaminate, si rivela essere un compito molto intricato.

Lo scopo del presente lavoro è quello di proporre due algoritmi in grado, rispettivamente, di discriminare immagini autentiche da immagini ritoccate e di individuare le zone delle immagini che sono effettivamente state modificate. Come fonte ispiratrice di questi algoritmi ci siamo basati su quelli vincitori della IEEE First Image Forensic Challenge, la prima competizione organizzata specificamente per rendere possibile il confronto fra metodi costituenti lo stato dell'arte dell'analisi forense di immagini. In particolare, il primo algoritmo è principalmente basato su un approccio di tipo machine learning, mentre il secondo consiste in una fusione fra due tecniche interamente nuove con una terza, ben più conosciuta nell'ambito.

I risultati, validati contro lo stesso dataset usato nella Challenge, mostrano che in entrambi i casi gli algoritmi proposti raggiungono o superano quelli vincitori della Challenge sul loro stesso dataset, allo stesso tempo lasciando aperte molte domande e possibilità di sviluppi futuri. Inoltre, come effetto collaterale di un'analisi del dataset mediante il primo degli algoritmi proposti, abbiamo individuato una nuova traccia forense che potrebbe essere sfruttata per orchestrare nuovi attacchi e che è già stata individuata inconsapevolmente dall'approccio vincitore della prima fase della Challenge.

Ringraziamenti

Innanzitutto devo ringraziare la mia famiglia per il loro costante supporto durante tutti questi anni. Ringrazio il mio relatore per avermi dato l'opportunità di svolgere questa tesi e di partecipare al progetto Rewind. Ringrazio anche Paolo per la preziosa assistenza e disponibilità durante tutte le fasi di questo lavoro.

Devo naturalmente ringraziare moltissimo Teo e Mara che mi sono sempre stati vicini e mi hanno fornito preziosi consigli (e divagazioni) sugli argomenti più disparati, in qualunque momento e luogo.

A questo punto dovrei anche citare un insieme $\mathscr{A}$ infinito (ma numerabile) di amici e persone conosciute nel corso degli anni, qui al Politecnico e non, che hanno contribuito anche loro a questa bellissima esperienza universitaria. Per evitare di citarli tutti, introducendo necessariamente una relazione d'ordine totale su $\mathscr{A}$ che non avrebbe motivo di esistere, li ringrazio tutti collettivamente senza citarne alcuno.

Contents

Chapter 1

Introduction

1.1 Motivations

Nowadays, the world is enormously influenced by the continuous availability of image and video content. Huge changes happened during the last dozen of years: one of the most significant is the source of such content. Until the 2000s, editors mostly relied on contributes produced by inside workers, such as journalists and professional photographers.

The appearance of smartphones and social networks changed the scene due to a variety of reasons. First, camera sensors are becoming incredibly sensitive in low-light situations: this enables users to easily capture images at night or in interior situations, thus documenting, for example, concert venues or parties. Smartphones are always readily available, as their multimedia capabilities can be accessed quickly with a few touches. Dedicated user friendly software applications enable advanced on-device processing, ranging from lighting correction to content removal. Expert users also may rely on ever more sophisticated programs built specifically to create forged multimedia content. Last but not least, smartphones are always connected to Internet, enabling users to instantaneously share produced content with the whole world through the usage of social networks.

Through the combination of these aspects, even a simple smartphone now can be considered as a powerful authoring tool. Bystanders became active producers of multimedia content, often being quicker than professional journalists. Many abuses and crimes became famous thanks to a Twitter post, or a Facebook status.

In the last decade, this has led us to a powerful deep change in the society: media have being used either to circumvent government censorship or to engage in it. This has brought unstable governments to their tipping

point, causing revolts and unrest. The known expression «A picture is worth a thousand words» summarizes the power of social media: implied information is delivered with much less effort than a written text, the delivery is instant and content can spread rapidly. This also means that altered content can diffuse much more easily than before, unsuspiciously conveying maliciously biased information to the general public. The need for instruments capable of distinguishing between altered and pristine multimedia content, is then more than an urgent necessity.

In scientific literature a number of techniques have been developed to this purpose, collectively known as "passive multimedia forensics". The common thread is that every technique exploits a trace left behind by any non-reversible operation which has been performed on the content. However, such traces are very difficult to detect, thus developed techniques either work only in restricted situations or involve some prior information on the content. Multimedia forensics also distinguishes between image [1] and video [2] content: in this work we will focus solely on the former.

Due to the increasing practical importance of forensic research, the IEEE Information Forensics and Security Technical Committee (IFS-TC) organized the First Image Forensics Challenge[1]. The Challenge is split into two parts: detection (*Phase* 1) and localization (*Phase* 2). The detection phase is aimed at evaluating the capabilities of tested algorithms to decide whether a given image is tampered or not, while the goal of the localization phase is to isolate tampered regions from pristine ones. The Challenge has ended, thus winners have already been proclaimed, and their winning methods have been released to the scientific community.

The goal of this work is to propose two new image forensics algorithms: one for image tampering detection, the other for image tampering localization. Since the Challenge is also split into tampering detection and localization, our algorithms are tested and validated on the same image set which has been used to train and test the submitted entries in the Challenge. As a consequence, this also results in a deep analysis of the challenge dataset.

In particular, for the first Phase we start from the machine-learning based approach which enabled the winners to achieve a very high score in the Challenge. We propose a way to improve it and we deeply analyze the reason why it works.

In the second Phase, instead, we propose a method which combines three different techniques, each one with its strenghts and its weaknesses. One of these techniques, PRNU, is well-known in the forensic literature, even though

[1] `http://ifc.recod.ic.unicamp.br/fc.website/index.py?sec=4`

we propose a new paradigm to exploit it; the other two, instead, are almost entirely novel. We also envision a way to combine said techniques in order to strenghten them reciprocally. This enabled us to achieve a better score than the one which has been obtained by the winners.

1.2 Thesis organization

The rest of work is organized as follows.

In Chapter 2 we introduce the necessary background in order to understand the proposed techniques. In particular, in Section 2.1 we first illustrate the image acquisition process in a digital camera: this is a conditio sine qua non for any image forensic method, since we will exploit traces left behind in some of the steps which any image undergoes during acquisition. After this, we introduce the Challenge, the rules and the accompanying dataset. In Section 2.3, we end the chapter with a description of the current state of the art on multimedia forensics, focusing especially on image forensics. We also detail the winning submissions since our algorithms will be built on those.

In Chapter 3 we describe the Detection problem (i.e. Phase 1). In particular we begin with a description of our proposed algorithm, first stating the theoretical setup in Section 3.1, then illustrating obtained results in Section 3.2. After having proven its effectiveness on the Detection problem, we further analyze the reason why it works: we do so in Section 3.3 by building additional datasets under different criteria, and using the same algorithm on those.

In Chapter 4 we describe the Localization problem (i.e. Phase 2). We begin in Section 4.1 by detailing each technique separately: we start with PRNU, and we finish with the two novel ones. We then proceed to discuss the theory behind the process of combining these techniques. In Section 4.2 we show the experimental results, mirroring the structure of the preceding section: first we report single techniques, then we discuss the fusion process. We finish also with a tentative adaptation of the algorithm described in the Detection phase, for the purpose of localizing altered regions.

In Chapter 5 we summarize what we did and the results we obtained. Since this work leaves some open questions, we propose a few directions in which next works could be oriented.

In Appendix A we briefly illustrate a machine learning classifier which is thoroughly used in Chapter 3.

To finish the work, in Appendix B we analyze the scoring mechanism of the Challenge and we report a log of every scores we obtained using the official

Challenge scoring mechanism. Finally we show some examples of tampered images which have been included into the Challenge dataset.

Chapter 2

Background

Multimedia forensics, despite being a very recent field, is very rich in literature: this is the research community response to the the ever-increasing availability of tools capable of modifying photographs without leaving any clue.

The goal of multimedia forensics is to guarantee proof of authenticity of a digital content: digital forensics techniques are able to detect whether a given content has been modified, which operations have been performed, where they have been applied and, in some cases, whether an alleged camera device has captured the given content or not. Multimedia forensics can be divided in two radically different branches: *active* forensics and *passive* forensics [1].

The active approach exploits pre-existing digital watermarks [3], purposely injected to detect and identify any modification occurred to the content. These watermarks can be either hidden or visible, and are robust to multiple destructive transformations. This approach is mainly followed by content producers who are in need to authenticate their work, such as musicians, photographers, police forces, etc. Simplicity and robustness are among its advantages, as the prior knowledge of the presence of a watermark is a very strong belief. The main disadvantage is that watermarks can only be inserted during the acquisition phase, thus requiring costly specialized instruments: for example, trustworthy cameras are equipped with watermarking chips which automatically sign each image with a private key, unique to that instance of a camera. This is infeasible, as it would require the development of a common standard among all camera makers, both in hardware and software requirements: for this reason, usage of trustworthy cameras is restricted to very limited scenarios.

The second branch does not rely on any purposely inserted watermarks, leveraging instead on recognizing traces (hereby *fingerprints*) left behind by

every operation which has been performed on the examined content. As a consequence, this branch is called *passive* multimedia forensics. In fact, every image or video is "altered" multiple times during its lifetime. The history of an image, or a video, can be roughly split into three stages: acquisition, coding and editing. Available passive multimedia forensics literature often target only a single stage at a time, without considering the authoring process as a whole. The passive approach is much more universal than active forensics, as it does not require any watermark. More often than not, this is the only viable approach to authenticate commonly available content. Since passive fingerprints are considerably weaker than digital watermarks, false tamper detection rate is much higher: therefore, the output of each passive forensic technique needs to be used to guide a forensic expert in deciding whether a given content is pristine or fake.

Passive multimedia forensic methods are often trained and validated using tools borrowed from the statistical world. Those tools employ large datasets of labeled media content, with varying degrees of prior knowledge; some datasets are built under fully controlled conditions, e.g. by shooting still scenes with reference objects. Others are fully blind, employing content downloaded from large public repositories such as Google or Flickr.

It is worth mentioning that research is being conducted in order to defeat currently available forensics methods: this research field, named *anti-forensics* or *counter-forensics*, leverages on the same weaknesses exposed in forensics literature. An anti-forensic method is thus aimed at hiding one kind of tampering from being uncovered by state-of-the-art detectors, perhaps by altering image statistics in order to match those who characterize natural images.

The aforementioned concepts on multimedia forensics can be applied to any kind of media. Indeed, multimedia forensics literature cover audio [4], image[1], and video [2] objects as well. In particular, since videos can be often regarded as sequences of images, video forensics inherit some tools from the image forensics field. Nonetheless, images and videos encoding schemes can be vastly different. As a consequence, some fingerprints can be reliably detected in both kinds of contents, such as lens aberrations, while others are hidden by the compression stage.

In this thesis, we focus solely on techniques for detecting and localizing tampering on images. To introduce the reader to the image forensics field we outline in the following sections some basic concepts. These include some background on digital image acquisition process, an overview of the image forensics algorithms more relevant to our work, and an introduction to the First Image Forensics Challenge organized by the IEEE, whose dataset has

been largely used in our work.

2.1 Image processing chain

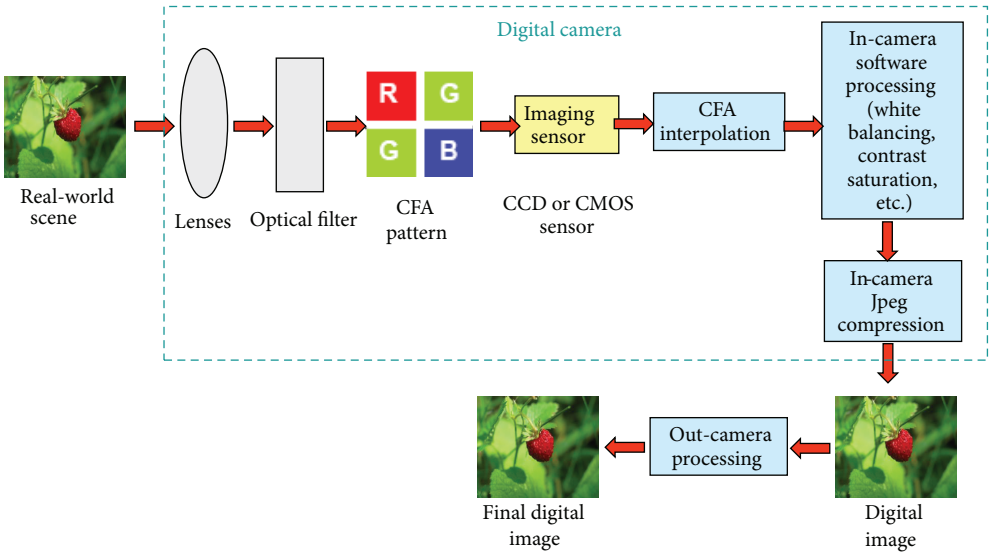


Figure 2.1: Common image processing chain. (Figure from [1])

With reference to Figure 2.1, in this section we analyze the typical processing chain used to generate a digital image. Understanding each step of this chain is essential for the development of forensic algorithms, since they leverage on traces left by processing operations to reconstruct the past history of an image.

During acquisition, light rays are shaped by the optical system and projected over the camera sensor. Lens imperfections can have a large impact on the resulting image quality, therefore a number of papers exploit aberrations introduced by spherical shape of the lenses [5].

Due to costs and lack of technology, pixels in common camera sensors are monochromatic. As a consequence, color information is captured by filtering incoming rays with a CFA (*Color Filter Array*), a thin transparent film which is patterned in a way such that each pixel captures only either the red, green or blue component. As a result, captured signals are therefore interpolated in order to reconstruct the full incoming color image.

An important factor in image sensors is the presence of noise during the acquisition phase: this term actually collects a multitude of unwanted contributes to the measured light values. Noise is commonly classified through

its temporal characteristics: realizations which do not change over time are called *fixed noise*, whereas the rest are called *random noise*. The former comprises isolated defects such as hot or stuck pixels, row and column-wise offsets in pixel response (FPN, *Fixed Pattern Noise*) and pixel-wise differences in light-to-value transfer (PRNU, *Photo Response Non Uniformity*), whereas the latter can be easily seen in low light situations.

Human perception is very different to what gets captured by a camera sensor: this is due not only to the differences between the human eye and a CMOS sensor, but also to the massive role of the human brain. In fact, optical illusions serve as examples of tricks which can be performed upon the human visual system. Their occurrence is actually quite common, but we do not consciously perceive them as our brain is adapted to their presence: a gray sheet of paper appears hueless under very different lighting conditions, such as sunlight or incandescent illuminants. However, the mediating mechanism is missing in camera sensors: a raw digital photograph of the aforementioned sheet often appears heavily tinted, and object colors can be very off from perceived ones. Hence, the role of the human brain is mirrored by on-camera processing, such as white balance, distortion correction, noise removal and gamma correction.

The processed image is often too large to be cheaply stored: by leveraging on human perception limits, a number of algorithms have been developed in order to compress the image data. Within the vast majority of consumer available cameras, this compression is performed directly on-board, and the resulting image is saved in the camera memory storage, often in JPEG format. The act of discarding imperceptible information has a price in forensics terms, as the compression algorithm can be often recognized. This stage is the most discriminant between image and video processing chains, as video compression greatly relies on the time axis to achieve very large compression gains.

At this stage of the chain, the image is finally made available to the user, who can perform at will a vast amount of actions with different aims, perhaps to enhance content rather than hiding an object from a photograph. A commonly used set of operations is cropping (a rectangular portion of the original image is kept while the rest is discarded), copy-move (a portion of the image is copied elsewhere), Healing Brush (a copy-move-like attack, where the copied content is automatically selected from the surroundings of the tampered region), splicing (a portion of an image is pasted inside another image) and resize (an image is scaled up or down using an interpolation algorithm). Additionally, an image can be saved multiple times during its lifetime, therefore the compression stage can be applied more than once, each time with

varying parameters and formats. Some examples of these attacks which have been carried out in the Challenge dataset can be found in Appendix B.3, in Figure B.2.

All the described operations (from lens effects to post-processing) leave peculiar footprints on the final image. Blind forensics methods exploit these footprints as an asset to detect which operations the image underwent during its lifetime. In the following section, we focus on these forensics algorithms.

2.2 The IEEE IFS-TC Challenge

Image forensics is a lively new field, and the practical significance of forensic research has induced The IEEE Information Forensics and Security Technical Committee (IFS-TC) to organize the First Image Forensics Challenge[1], started in June 2013.

The purpose of the challenge is multiple: first, forensics research lacked a clear benchmark dataset to compare on, relying instead on proprietary sources and hand-made forgeries. Secondly, the quest served as an opportunity for digital forensics researchers to showcase their developed algorithms and to sharpen their skills in a real-world testbed. Finally, it suggests for a common standardization protocol as a common comparison ground truth for new techniques.

The challenge consists of two phases: detection (*Phase* 1) and localization (*Phase* 2). The detection phase is aimed at evaluating the capabilities of tested algorithms to decide whether an image is tampered or not, while the goal of the localization phase is to isolate tampered regions from pristine ones.

A set of various natural images is provided, split into a labeled training set and a unlabeled test set (as shown in Table 2.1). The former suffices both for detection and localization, while the latter is itself split between the two phases: in particular, all images in Phase 2 test set have been tampered with. Each training image is paired with a black & white mask, showing which regions have been modified, as shown in Figure 2.2: a fully white mask declares that the image is pristine. More samples are reported in Section B.3.

Tampered images have all been created by different attackers from all over the world (i.e. from China to Brazil), using different kinds of forgeries. This ensures both that the dataset is unbiased and that it contains many different examples of forgeries commonly found in a real-world scenario. It is worth noticing that the dataset has been created taking into account some well known image forensics methods. To this purpose, forged images have

[1] `http://ifc.recod.ic.unicamp.br/fc.website/index.py?sec=4`

been processed in order to fool many state-of-the-art detectors (e.g., those exploiting JPEG compression footprints), pushing the forensics community to propose novel techniques.

Phase	Set	Image count		
		Fake	Pristine	Total
Detection and Localization	Training	450	1050	1500
Detection and Localization	Test	?	?	5713
Localization	Test	700	0	700
Total				7913

Table 2.1: Challenge dataset composition. Question marks indicate that some data has not been publicly released.

(a) Fake image *(b) Corresponding mask*

Figure 2.2: A sample training image/mask pair, showing that the bar stool has been added later.

Each challenger competes by submitting his full guesses on the nature of one of the two test sets: a score is computed daily for each submission. To prevent oracle attacks, the score is computed on a randomized subset of the submitted masks.

Actually, the challenge has ended, and the winners have already been proclaimed, both in phase 1 and 2. However, it is still possible to submit guesses in the both phases, as the test set ground truths have not been released yet: the submission mechanism will be used in this thesis to evaluate the performance of the examined algorithms, whereas the winner algorithms have been analyzed in detail.

As a side note, 8 of the 450 masks in the Challenge training set are defective, rendering the images unusable for Localization training: still, they can

be used for Detection training.

It is worth noticing that the challenge also allows the use of metadata (such as file header information) as a clue for tampering detection. However, results obtained with these methods are not ranked together with those obtained using solely signal processing techniques. Since we focus on signal processing, we do not take into account other methods. Notice that using header information, a participating team was able to perfectly detect tampered images into the test set (i.e., 100% accuracy). This means that the ground truth for Phase 1 test set has been somehow revealed: as a consequence, the ground truth labels in $\mathcal{I}^1_{\text{ts}}$ have been released aside from the Challenge, hence the analysis will extend also on the test set to discuss generalization error, without exploiting it in any way to improve existing techniques or to devise others.

2.2.1 Scoring rules

The Challenge is a binary classification problem, therefore scoring is computed from the contingency matrix between ground truth and the submitted guesses. The Challenge phases employ different scoring systems. Let $\mathscr{S} \subset \{1,\dots,|\mathcal{I}|\}$ be a set of images whose the score will be computed on.

To prevent oracle attacks, scores are actually computed using a randomized subset (i.e. 70%) of $\mathscr{S}$: for now, we neglect the randomization. We note that its effect on the resulting score has been empirically analyzed in Appendix B.1.

Phase 1

Phase 1 score S_1 is computed using ground truth labels $l_i \in \{0,1\}$ associated to each i-th image $i \in \mathscr{S}$, where "1" stands for "fake", and "0" marks a pristine image. Let $\widehat{l_i}$ be the user's guess on the label associated to i-th image. In this context, we define a set of quality measures as

$$\begin{aligned} TP &= \left|\left\{i \in \mathscr{S} : l_i = 1 \wedge \widehat{l_i} = 1\right\}\right| & TN &= \left|\left\{i \in \mathscr{S} : l_i = 0 \wedge \widehat{l_i} = 0\right\}\right| \\ FP &= \left|\left\{i \in \mathscr{S} : l_i = 0 \wedge \widehat{l_i} = 1\right\}\right| & FN &= \left|\left\{i \in \mathscr{S} : l_i = 1 \wedge \widehat{l_i} = 0\right\}\right| \\ P &= \left|\{i \in \mathscr{S} : l_i = 1\}\right| & N &= \left|\{i \in \mathscr{S} : l_i = 0\}\right|. \end{aligned} \tag{2.1}$$

Phase 1 score S_1 is computed as

$$S_1 \coloneqq \frac{1}{2}\frac{TP}{P} + \frac{1}{2}\frac{TN}{N} \tag{2.2}$$

.

Phase 2

Phase 2 score is computed in two phases and evaluates, instead, the user submitted masks $\{\widehat{M}_i\}_{i\in\mathscr{S}}$ against the ground truth masks $\{M_i^{\mathrm{GT}}\}_{i\in\mathscr{S}}$.

First, a score $S^{(i)}$ is computed for each i-th image $i\in\mathscr{S}$. Let $M_k\in\{0,1\}$ be the ground truth value associated to k-th pixel in mask M_i^{GT} and let $\widehat{M}_k$ the user's guess on M_k. Let $\mathcal{K}$ be the set of pixels in mask M_k. Akin to Equation 2.1, for each image $i\in\mathscr{S}$ we define

$$\begin{aligned} TP &= \left|\left\{k\in\mathcal{K} : M_k = 1 \wedge \widehat{M}_k = 1\right\}\right| & TN &= \left|\left\{k\in\mathcal{K} : M_k = 0 \wedge \widehat{M}_k = 0\right\}\right| \\ FP &= \left|\left\{k\in\mathcal{K} : M_k = 0 \wedge \widehat{M}_k = 1\right\}\right| & FN &= \left|\left\{k\in\mathcal{K} : M_k = 1 \wedge \widehat{M}_k = 0\right\}\right| \\ P &= \left|\left\{k\in\mathcal{K} : M_k = 1\right\}\right| & N &= \left|\left\{k\in\mathcal{K} : M_k = 0\right\}\right|, \end{aligned} \tag{2.3}$$

where, for example, FP is the total number of pristine pixels which have been classified as being tampered. With these definitions, for the i-th image we compute the F_1 score:

$$S^{(i)} := \frac{2TP}{2TP+FP+FN} \tag{2.4}$$

As an example, suppose that $M_i = M_i^{\mathrm{GT}}$ is fully white (i.e. the image is pristine), and the user submitted mask is half white and half black (i.e. the image is classified as being half pristine and half fake). Then, $TP = FN = P = 0$ (i.e. there are no fake pixels), $N = |\mathcal{K}|$ (i.e. all pixels are pristine), and $TN = FP = \frac{|\mathcal{K}|}{2}$ (we misclassified half part of figure).

Finally, the overall Phase 2 score S_2 is the arithmetic mean of all $S^{(i)}$ as in

$$S_2 := \frac{1}{|\mathscr{S}|}\sum_{i\in\mathscr{S}} S^{(i)} \tag{2.5}$$

Further metrics

To evaluate the performances of described methods, using the definitions we gave in this Section we can also define other useful metrics: we summarize them in Table 2.2.

Name	Definition
True Positive Rate	$TPR \coloneqq \frac{TP}{P}$
True Negative Rate	$TNR \coloneqq \frac{TN}{N}$
False Positive Rate	$FPR \coloneqq \frac{FP}{P}$
False Negative Rate	$FNR \coloneqq \frac{FN}{P}$
Accuracy	$ACC \coloneqq \frac{TP+TN}{P+N}$
F_1	$F_1 \coloneqq \frac{2TN}{2TN+FN+FP}$

Table 2.2: Statistics based on a binary contingency matrix.

2.3 State-of-the-art on image forensics

As stated in this chapter, different kinds of forensic methods are available to deal with different media (i.e. audio, images, and video) in different ways (i.e. passive or active). Since the goal of this work is to analyze images in a real-world scenario, we cannot assume the usage of any fingerprinting or watermarking method: as a consequence, we focus solely on passive image forensic methods.

Blind image forensics methods exploit traces left by different processing operations. Each trace is the proof that the image underwent a given processing operation. Depending on the operations that are detected, an image can be considered forged or pristine. In case these traces are extracted locally, rather than globally on the whole image, it is also possible to localize the possible forgery in the pixel domain. In the following, we give a brief overview of some of the many traces that can be detected. Later on, we will focus on the traces that we actually exploit in depth in this thesis.

Following the image processing chain (Figure 2.1), light rays are first captured by the camera optics, shaped by its imperfections and projected onto the camera sensors. Chromatic aberrations can be used to localize image tampering [6] or to identify mobile phones [7]. Purple fringing can also be used for tamper detection [8]. Since DSLR (Digital Single Lens Reflex cameras) are vulnerable to the presence of dust particles on the sensor, they can be used as a means to identify the used camera [9].

The color of each pixel is captured by letting the light shine trough a CFA matrix: the resulting grid of monochromatic pixels is then interpolated in order to reconstruct the full color information. Interpolation algorithms and CFA structure may differ between camera models and camera manufacturers, thus a weak form of source identification is possible [10], [11]. Moreover,

interpolation artifacts may uncover the presence of a local tampering [12], [13]. A drawback of these methods is the large block size requirement: local tampers smaller than the sliding block cannot be detected.

CMOS sensors are greatly affected by the presence of noise: PRNU constitutes a significant non-random, temperature independent and stable contribute over a camera lifetime. Due to these characteristics, it is often targeted by forensics research; PRNU-based forensics methods can be used both to detect local forgeries, where the PRNU pattern is altered, and to assign an image to a camera. PRNU estimation usually starts from a set of images which have been taken by the targeted camera [14], whereas camera attribution is carried out by correlating a known fingerprint with the image noise [15]. It is also possible to recover camera parameters such as scaling and digital zoom [16]. PRNU methods can be applied to printed images as well [16]. In [17], PRNU is coupled to CFA for camera model/brand identification.

After acquisition, the image is often compressed using the industry-standard JPEG algorithm [18]. The JPEG pipeline starts with downsampling the color information present in the raw input image, as our eye is less capable of distinguishing color details than intensity variations. The raw downsampled image is then partitioned into a grid of 8×8-sized blocks. Each block is then transformed into frequency domain through the Discrete Fourier Transform. As the highest frequencies often comprise noise and smaller imperceptible details, each block is entry-wise divided by a given quantization matrix Q, rounded up (or down), and multiplied by the same Q: this is called the *quantization* step. Finally, the image is fed to a run-length encoder and stored into memory.

The quantization step is the sole lossy step of the JPEG pipeline: in fact, JPEG compression produces distinctive artifacts on the boundary of each 8×8 block. Moreover, the quantization matrix Q is not specified by the standard but its choice is left open to the implementer. JPEG forensics methods can be divided in two branches: those which analyze the structure of Q and those who do not, relying instead on detecting inconsistencies across neighbouring blocks, such as [19]. Among the first branch, we distinguish methods aimed at detecting whether a JPEG image has been compressed more than once (the so-called *double JPEG*, or DJPEG). As the first digit of quantized coefficients in natural images follows the Benford law, this fact is exploited in [20]. Available methods also separate the case which the latest JPEG compression has been performed using the same grid as the preceding one (*aligned* DJPEG) or not (*not aligned* DJPEG), such as [21].

At this step of the chain, the image can be edited by the user through a multitude of editing operators. Forensic methods often target specific op-

erations, which may span from malicious alterations, such as copy-move or splicing, rather than simple image enhancement, such as increasing contrast or removing noise.

The seminal paper for copy-move detection is [22], in which the image is partitioned into blocks which will be clustered together according to their similarity and their reciprocal distance. Large clusters of similar blocks which are displaced by the same relative amount, reveal the presence of a copy-move attack. Among these forensic methods, a very effective algorithm is PatchMatch [23]: common tools for content removal (such as Adobe Photoshop healing brush) actually make use of PatchMatch, thus establishing its status both as a forensic and an antiforensic method. Another approach for copy-move detection relies on local descriptors obtained through the scale-invariant feature transform (SIFT): instead of matching similar blocks in an image, *keypoints* are clustered together in order to identify similar areas in a picture. [24]

Geometrical transformations are not malicious per se, but they can still be used as a mean to distort the real size of an object. As such, the attacked image (or a part of it) is subject to resampling, thus introducing spurious correlations between image pixels. Resampling detectors work by revealing such correlations [25], or anomalous periodicities in the interpolated image. [26].

Splicing detection, however, is much more difficult than copy-move detection, although it may seem related. It can be carried out through CFA or PRNU-based methods, as it is done in this thesis, or by dedicated content-based methods. The human brain is weak at inferring the correct physical and geometrical properties of a scene, therefore even a skilled attacker may be able to craft a fake image which achieves a convincing look, but still being geometrically inconsistent. Hence, most physical-based methods target elements which are particularly difficult to reproduce realistically. However, the forensic process often needs to be human-assisted, thereby rendering it inapplicable on large amounts of data.

The main targets for content-based forensic methods are lighting, shadows and perspective. At present, the lighting direction of an object in an image cannot be modified, therefore a careless splicing produce inconsistency in illuminant direction, color, number of lights and cast shadows. The first is addressed in [27] under a number of simplifying hypotheses, such as restricting the analysis to Lambertian surfaces on the sides of the scene objects. The presence of multiple lights sources is exploited in [28], while their differences in colors are studied in [29]. More interestingly, a number of methods have been proposed in [30] and [31] which exploit shadow size, direction and photometric properties, under single (outdoor scenes) or multiple light sources

(indoor scenes). Perspective consistency is also an important forensic tool: the uniqueness of the vanishing point of a scene is easily violated by fake images, thus enabling recognition of spliced images using, for example, human eyes [32] or text [33].

A very effective antiforensic method is the so-called *recapture*, which consists in taking a photograph of a reproduction of the altered photograph on a printed media or a screen. As most forensic techniques are bound to recognizing inconsistencies with respect to the camera which took the said photograph, a recaptured image fails to be detected. Still, some geometrical clues can be exploited [34] rather than specularities in printed images [35] or a combination of features [36].

2.3.1 On PRNU

Since in this thesis we make largely use of PRNU, we decided to introduce a more in depth state-of-the-art analysis of PRNU-based image forensics algorithms. In particular we focus on PRNU estimation and the usage of PRNU both for camera identification and for tampering localization.

Estimation

Stating from [14] and considering a single color channel, PRNU associated to the pixel with coordinates (i, j), can be modeled as a multiplicative zero mean noise $K(i, j)$ acting on an ideally noiseless image pixel $I_0(i, j)$, whereas other noise sources (e.g. shot noise, quantization noise, dark currents, ...) are described with a spatially uncorrelated noise process $\Theta(i, j)$. Dropping the pixel indices, the image model is the following:

$$I = I_0 + K I_0 + \Theta \tag{2.6}$$

The estimation of K is problematic, the term $K I_0$ cannot be present in regions where the incoming light has saturated the sensor (i.e., I is close to the maximum admissible value, 255 for 8-bit images) [37], and its magnitude is usually negligible with respect to Θ, especially in dark regions where I_0 is small.

PRNU extraction requires the possession of a set $\mathscr{S}$ of N unprocessed images

$$\mathscr{S} := \left\{ I_k \right\}_{k=1}^{N}$$

taken by the same camera. In principle, estimation of K can be performed simply by weighted averaging. To aid PRNU extraction, image content is

suppressed by subtracting from Equation (2.6) an estimate of I_0: such estimate $\widehat{I_0}$ is typically obtained with a denoising filter $\mathcal{D}$, while W is the noise residual.

$$W := I - \widehat{I_0} = I - \mathcal{D}(I) \tag{2.7}$$

In [38] and [39], the performance of several denoising filters $\mathcal{D}$ is compared. In this thesis we use the Mikçak filter first described in [40]: this is a simple adaptive spatial filter which is aimed at removing additive white Gaussian noise, assuming the knowledge of the noise variance σ^2. As suggested by [14] and considering that the challenge images are similar both in content and in nature with those examined in the paper, σ^2 is set to 3.

Other available filters include BM3D [41], which exploits characteristics typically shown by natural images. First, an image is decomposed into a set of small blocks, from which a sparse representation is learned. Next, groups of blocks are matched against each other: components shared by every block in a group are enhanced, thereby enhancing also small recurrent details, while others, such as noisy contributes, are shrinked. Finally, the processed blocks are fused using a spatial collaborative Wiener filter. The obtained non-parametric filter exhibits state-of-the-art performance.

Equation (2.7) can be further rewritten as

$$W = I - \widehat{I_0} = I_0 + KI_0 + \Theta - \widehat{I_0} = \left(I_0 - \widehat{I_0}\right) + K(I_0 - I) + KI = KI + \Xi\,, \tag{2.8}$$

where Ξ is a term which collects multiple noise contributes and reconstruction error introduced by the denoising filter. Let us assume that image content is successfully suppressed in W: if I is a photograph of a well-exposed smooth scene like a cloudy sky, it's reasonable to assume that Ξ can be modeled by a white Gaussian process with 0 mean and variance σ^2, since the skewing noise components (e.g. dark currents) are negligible. Moreover, Ξ can be assumed as being independent from KI [14].

Under these assumptions, the maximum likelihood estimator for K from a set of N images $\{I_k\}_{k=1}^N \in \mathscr{S}$ (with corresponding noise residuals $\{W_k\}_{k=1}^N$) can be derived. The likelihood is

$$\mathscr{L}\left(K \mid \{I_k\}_{k=1}^N, \{W_k\}_{k=1}^N\right) = -\frac{N}{2}\sum_{k=1}^{N}\log\left(\frac{2\pi\sigma^2}{(I_k)^2}\right) - \sum_{k=1}^{N}\frac{\left(\frac{W_k}{I_k} - K\right)^2}{2\sigma^2/(I_k)^2}$$

By maximizing the likelihood, the ML estimator for the PRNU pattern K can be written as

$$\widehat{K} := \arg\max_K \mathscr{L}\left(K \mid \{I_k\}_{k=1}^n, \{W_k\}_{k=1}^N\right) = \frac{\sum_{k=1}^N W_k I_k}{\sum_{k=1}^N I_k^2} =: \sum_{k=1}^N m_k W_k$$

The form of $\widehat{K}$ is consistent with pixel-wise averaging noises W_k over the set $\mathscr{S}$, weighted by the corresponding pixel normalized intensities m_k. It is experimentally shown that K also captures contributes not strictly related to PRNU, such as periodic patterns induced by the CFA structure, row-wise and column-wise offsets produced by the scanning mechanism of the camera sensor, and other artifacts which depend on various factors such as exposure time, focal length, lens, camera model [42]. As those contributes do not lead to a unique camera, they are usually removed using a composition of Wiener filtering in frequency domain and removal of column/row means [14]. Moreover, pixels approaching saturation (e.g. close to the maximum possible value) do not convey any information on PRNU, therefore a set of corrected weights $\tilde{m}_k$ is also used. For 8-bit grayscale images, PRNU clipping is dealt by defining

$$\tilde{m}_k = \begin{cases} m_k & \text{if} \quad I_k < 252 \\ 0 & \text{if} \quad I_k \geq 252 \end{cases} \tag{2.9}$$

whereas filtering is performed as follows:

$$\widehat{K}^0 := \mathscr{F}^{-1}\left\{\mathscr{F}\left(\mathscr{Z}\left(\widehat{K}\right)\right) - \mathscr{W}\left(\mathscr{F}\left(\mathscr{Z}\left(\widehat{K}\right)\right)\right)\right\} \tag{2.10}$$

where $\mathscr{F}$ denotes the Discrete Fourier Transform, $\mathscr{Z}$ is the said operator which removes means from each row and column, and $\mathscr{W}$ is the 3x3 Wiener filter with variance estimated using the image-wise sample variance of the k-th noise residual $\left\{W_{k(i,j)}\right\}_{i,j}$. From now on, for the sake of simplicity, the symbol $\widehat{K}$ will denote the filtered PRNU $\widehat{K}^0$.

Camera attribution

Let K be a camera PRNU fingerprint, previously estimated from known pristine images. Let I be an image whose camera is unknown. Let $\overline{K}$ be a PRNU pattern of a different camera. First, image noise W is extracted from I as in Equation (2.7). Camera attribution is modeled within hypothesis testing framework as a decision problem between two different noise models:

$$\begin{cases} \mathrm{H}_0: & W = \overline{K}I + \Theta \\ \mathrm{H}_1: & W = KI + \Theta \end{cases}$$

where Θ is a white Gaussian noise with 0 mean. The alternative hypothesis H_1 holds when the camera is correctly identified by the given PRNU K, while the null hypothesis H_0 holds when either PRNU is not detected (e.g. the image has been tampered or belongs to a camera with a PRNU pattern $\overline{K} \neq K$).

Since the term $\overline{K}I$ is usually negligible with respect to Θ [14], the hypothesis testing problem can be reformulated as follows:

$$\begin{cases} \mathrm{H}_0: & W = \Theta \\ \mathrm{H}_1: & W = KI + \Theta \end{cases} \tag{2.11}$$

In order to write the test statistics for Equation (2.11), let us define a few operators.

A dot product $\cdot$ between $N \times M$ pixel images can be defined, along with its induced norm $\|\cdot\|$ in the image space:

$$A \cdot B \coloneqq \sum_{i=1}^{N} \sum_{j=1}^{M} A(i,j)B(i,j) \qquad\qquad \|A\| \coloneqq \sqrt{A \cdot A}$$

Moreover, the sample mean can also be introduced:

$$\overline{A} \coloneqq \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} A(i,j).$$

Borrowing from the stochastic processes theory, an image $A = \{A(i,j)\}_{i,j}$ can be viewed as a finite set of realizations of an infinite process. By applying zero-padding, one may define convolution and cross-correlation between two images A and B:

$$C = C(i,j) = (A * B)(i.j) \coloneqq \sum_{m,n=-\infty}^{+\infty} A(m,n)B(i-m, j-n)$$

$$C = C(i,j) = (A \otimes B)(i.j) \coloneqq \sum_{m,n=-\infty}^{+\infty} A(m,n)B(i+m, j+n).$$

The cross-correlation range can be restricted to the closed interval $[-1,1]$. The resulting operator is called *normalized cross-correlation*:

$$C = C(i.j) = \mathrm{NCC}[A,B](i,j) \coloneqq \frac{\left((A-\overline{A}) \otimes (B-\overline{B})\right)(i,j)}{\left\|A-\overline{A}\right\| \left\|B-\overline{B}\right\|}. \tag{2.12}$$

Using the Generalized Likelihood Ratio Test, the test statistics for (2.11) is the correlation ρ between IK and W:

$$\rho \coloneqq \mathrm{NCC}[IK, W](0,0). \tag{2.13}$$

H_0 is rejected when ρ is lower than a correlation threshold γ. Experimental values for γ vary around 0.01 [37], and may decrease or increase due to cropping, scaling or further processing. In Figure 2.3 we show a sample task of detecting images taken with one camera against five other cameras.

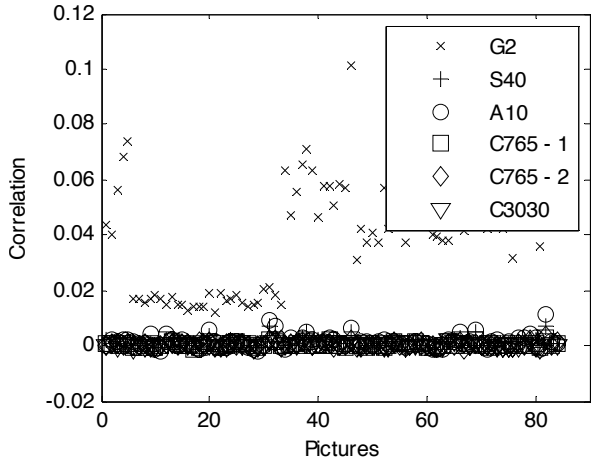


Figure 2.3: Values of ρ obtained by comparing 84 JPEG images taken by a Canon G2 with six PRNU patterns, five of which correspond to other cameras. Notice how well a the correlation threshold γ to 0.01 discriminates images taken with the Canon G2, whose pattern has been extracted, from images taken with other cameras. (Figure from [37])

In the PRNU-based forensics context, $\mathrm{NCC}[A,B](i,j)$ is an often sharp-peaked function in the shift arguments i and j: this is mainly due to the fact that IK and W are almost-everywhere uncorrelated if their relative position is shifted from each other than the true value, e.g. whenever the query image I has been cropped with respect to the full images used to estimate K. As a consequence, rather than simply analyzing ρ, it is more effective to detect whether we are approaching a peak.

In [43], the *Peak to Correlation Energy ratio* (PCE) is introduced as a useful measure of peakness. Let $(i_{\mathrm{peak}}, j_{\mathrm{peak}})$ be the argmax of $\mathrm{NCC}[IK,W](i,j)$, and let $\mathcal{N}$ be a small neighborhood around such maximum. The PCE is defined as follows:

$$\mathrm{PCE}(A,B) := \frac{\left(\mathrm{NCC}[A,B](i_{\mathrm{peak}}, j_{\mathrm{peak}})\right)^2}{\frac{1}{NM-|\mathcal{N}|}\sum_{i,j\notin\mathcal{N}}\left(\mathrm{NCC}[A,B](i,j)\right)^2}$$

As for NCC, a threshold τ is usually set on PCE, depending on the desired significance level and the test power.

Tamper localization

The same attribution framework can also be applied by correlating N blocks extracted from W and K, say W_b and K_b, $b \in \{1,\ldots,N\}$, to the purpose of un-

covering local tampers whenever K_b is not detected. This is the preferred approach in image forensics literature for PRNU-based methods. However, more hypotheses are required, since a correspondence between W_b and K_b needs to be established first: for example, processing involving scaling and other geometrical transformations need to be ruled out.

The output of a localization approach is a correlation surface $R = R(i_b, j_b)$, where (i_b, j_b) are the pixel coordinates of the center of block b. Let W be the denoising residual as in Equation (2.7). Suppose that no cropping is performed: that is, the query image I has the same shape as the reference pattern K. As a consequence, following Equation (2.13), each residual block W_b can be correlated with the corresponding product $I_b K_b$, thus giving rise to a test statistic for each block b:

$$R(i_b, j_b) := \rho_b = \mathrm{NCC}\left[I_b K_b, W_b\right](i_b, j_b) \qquad \forall b \in \{1, \ldots, N\} \tag{2.14}$$

By applying the same thresholding procedure as in 2.3.1, a binary mask is obtained which separates fake regions from pristine ones.

Similarly, we may compute the PCE between each block, obtaining a PCE surface $P = P_b$:

$$P_b := PCE\left(I_b K_b, W_b\right) \qquad \forall b \in \{1, \ldots, N\} \tag{2.15}$$

Several variations on (2.11) have been proposed: [14] allows for small deviations from the model in Equation (2.13), and builds a predictor for K on regions where H_0 is *not* rejected. In [44], the threshold γ is chosen using the Bayesian decision theory.

2.3.2 Winning algorithms

Several groups have taken part in the Challenge. The submissions which won both phases of the Challenge were both made by Team GRIP[2] from Università Federico II di Napoli. High scores have been achieved on both phases: see Table 2.3a and Table 2.3b for the final ranking of the participants in both phases.

The winners were required to submit a 4-page paper which detailed their approach. Since these papers are the most up to date state-of-the-art algorithms for tampering detection and localization, we outline the core of these algorithms that also serve as starting point for some techniques developed in this work.

[2] `http://www.grip.unina.it/`

Place	Leader	Team	Score
1	**Luisa Verdoliva**	**grip**	**0.942064**
2	Guanshuo Xu	havefun	0.937259
3	xinqi lin	hryup	0.934629
4	Licong Chen	Chen	0.932270
5	Khosro Bahrami	Fake Bluster	0.857373
6	Dev Sh	ITD	0.823980
7	Han April	April	0.570658
8	Anh-Dung LE	Buster	0.542519

(a) Phase 1: detection

Place	Leader	Team	Score
1	**Luisa Verdoliva**	**grip**	**0.407172**
2	Guanshuo Xu	havefun	0.267773
3	Licong Chen	Chen	0.184282
4	xinqi lin	hryup	0.164267

(b) Phase 2: localization

Table 2.3: Challenge final scores, computed according to 2.2.1.

Some notation

Let $\mathcal{I}$ be the full image set of the Challenge. Following Table 2.1, $\mathcal{I}$ is split in a training set $\mathcal{I}_{\mathsf{tr}}$, whose image ground truths (from now on, *labels*) are known, and a test set $\mathcal{I}_{\mathsf{ts}}$, on which the Challenge scores are computed. $\mathcal{I}_{\mathsf{tr}}$ can be itself split into $\mathcal{I}_{\mathsf{tr,P}}$ and $\mathcal{I}_{\mathsf{tr,F}}$. The cardinalities of each set are reported in Table 2.1. In Figure 2.4 we represented graphically the established notation along with the Challenge dataset.

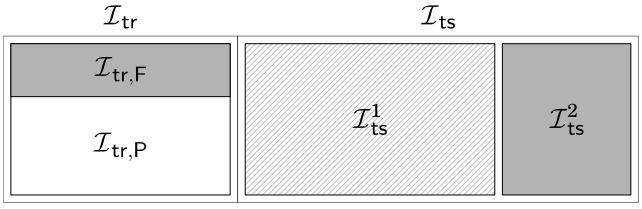


Figure 2.4: The Challenge dataset composition (not to scale): gray areas are associated to fake images.

Let $l_i \in \{0, 1\}$ be the ground truth label associated to the image $i \in \mathcal{I}$, where the value 0 is associated to "*pristine*" status.

Detection

Team GRIP achieved a very high score in the detection phase, using a promising new approach which involved machine learning tools trained on features originally envisioned in the steganalysis field, further augmenting their score using the PatchMatch algorithm. Their results have been published and are currently available on `ArXiv` [45]. Moreover, due to the large amount of features and the small size of the training dataset, they heavily rely both on cross-validation and ensemble learning in order to obtain robust predictions.

The starting point of their approach is steganalysis based, and involves the computation of co-occurrences in a noise-like matrix. Following [46], a vector of 39 groups of features (from now on, *submodels*) is extracted for each image, while each submodel is comprised either by 328 or 335 values in $[0,1]$. The extraction procedure starts with converting each image in grayscale mode: the resulting image will be denoted as $I(i,j)$.

Next, Cozzolino, Gragnaniello, and Verdoliva following the approach described by Fridrich and Kodovsky, built a set of 39 high-pass-like filters acting on each one of the $I(i,j)$. Let $\mathcal{N}(i,j)$ be a small punctured disk around pixel (i,j), and let $\widehat{I}[A](i,j)$ be a prediction of $I(i,j)$ using pixels in set A. Then, the filtered image $r(i,j)$ can be expressed as

$$r(i,j) \coloneqq \widehat{I}\left[\mathcal{N}(i,j)\right](i,j) - cI(i,j),$$

where c and the predictor depend on the used filter. The relationship between the predictor and c is shown in Figure 2.5, whose color/number/shape code is translated to the type of the operation which is performed on each pair of pixels. Available filters include differences between neighbours and non-linear operators such as max and min:

$$\begin{aligned} r(i,j) &= X(i,j+1) - X(i,j) \\ r(i,j) &= \min\left[\left(X(i,j+1) - X(i,j)\right), \left(X(i+1,j) - X(i,j)\right)\right]. \end{aligned}$$

The computed residuals $r(i,j)$ are quantized and truncated:

$$\bar{r}(i,j) \coloneqq \left(\mathrm{Trunc}_T \circ \mathrm{Round}\right)\left(\frac{r(i,j)}{q}\right) \tag{2.16}$$

where $T \in \mathbb{N}$ and $q > 0$ are a values of choice, $\mathrm{Round}(x)$ rounds x to the nearest integer and Trunc_T is the truncation function:

$$\mathrm{Trunc}_T(x) \coloneqq \begin{cases} x & \text{if } |x| < T \\ T\ \mathrm{sign}(x) & \text{if } |x| \geq T \end{cases}$$

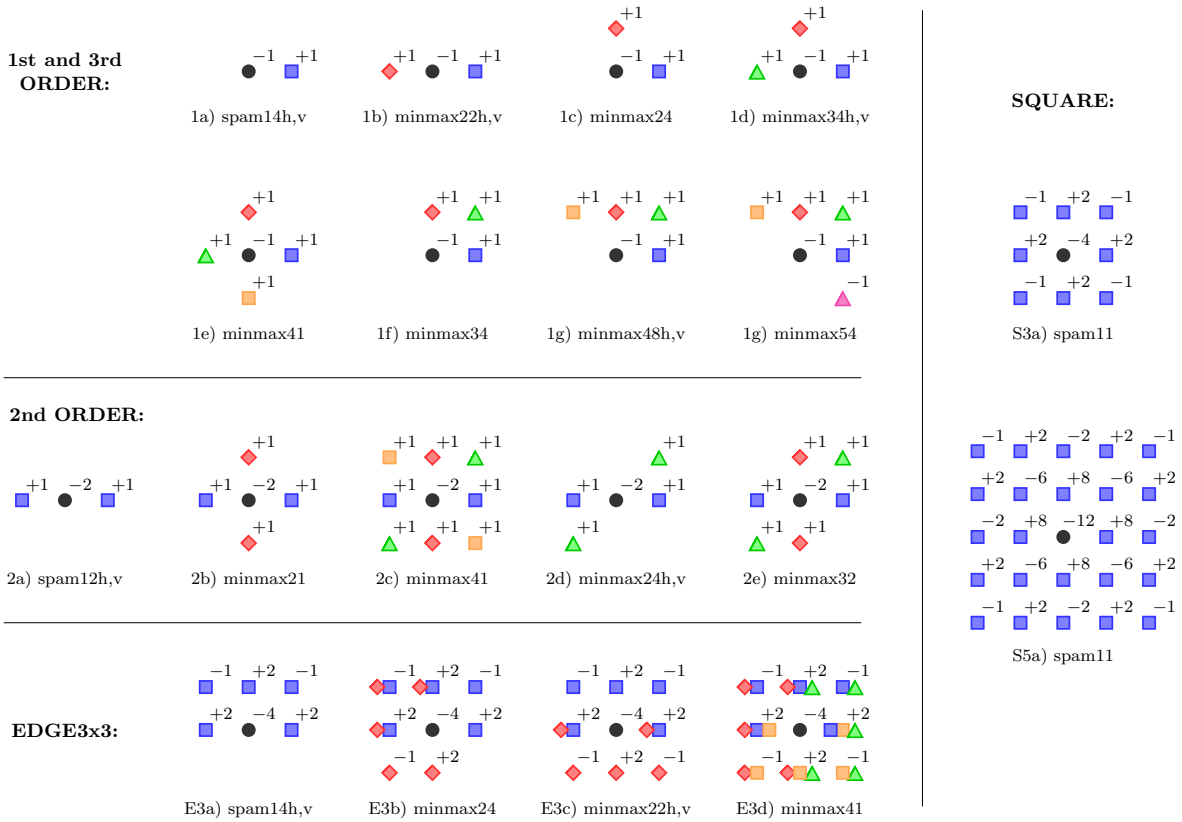


Figure 2.5: Stencils of the high-pass filters. Further details in [46]

As a result, $\{\bar{r}(i,j)\}_{i,j}$ is a matrix with the same shape as the original image, but whose entries take values in the set $-T,\dots,T$. In the examined articles, T always takes value 2 due to excessive computational burden, while q is varied in [46] but fixed to 1 in [45] to avoid overfitting.

Next, co-occurrence matrices are formed starting from the $\{\bar{r}(i,j)\}_{i,j}$ produced by each type of submodel. The i-j-th entry of a co-occurrence matrix counts how many times one given string of values of $r(i,j)$ appears adjacent to another one. The length of the string is fixed: in [46] and in the winning Challenge entry [45], the Authors experimented with 4 neighbours, but further analyses are encouraged, while the strings are constrained to have horizontal or vertical directions. As a result, each co-occurrence matrix has 25×25 entries, where 25 is the number of possible combinations of 4 consecutive values which can be found in $r(i,j)$ under the previous hypotheses. This is a large number when compared with the amount of available images, but stencil symmetries can be exploited to reduce the obtained degrees of freedom from 625 to 328 or 325 depending on the submodel. Finally, the resulting co-occurrence

matrices are put in a vector form, concatenated together and resulting vector is set to sum to 1.

Up to this point, Cozzolino, Gragnaniello, and Verdoliva managed to train a supervised *Support Vector Machine* linear classifier[3] on each submodel, independently from the others, heavily relying on cross-validation. The Challenge training set $\mathcal{I}_{\text{tr}}$ is split at random into a cross-validation training set with unbalanced classes, on which the SVM are trained, and a cross-validation test set. The cross-validation scheme is represented in Figure 2.6. Following notation described before

$$\begin{aligned} \mathcal{I}_{\text{tr}} &= \mathcal{I}_{\text{tr}}^{\text{CV}} \cup \mathcal{I}_{\text{ts}}^{\text{CV}} \\ \left|\mathcal{I}_{\text{tr}}^{\text{CV}} \cap \mathcal{I}_{\text{tr,F}}\right| &= {}^{5}/_{6} \left|\mathcal{I}_{\text{tr,F}}\right| \\ \left|\mathcal{I}_{\text{tr}}^{\text{CV}} \cap \mathcal{I}_{\text{tr,P}}\right| &= {}^{5}/_{6} \left|\mathcal{I}_{\text{tr,P}}\right| \end{aligned}$$

The m-th classifier is trained on the cross-validation training features associated to the m-th submodel along with the image ground truths $\{l_i\}_{i \in \mathcal{I}_{\text{tr}}}$ as supervised outputs; to each classifier, a cross-validation phase 1 score is computed on the labels predicted by feeding the cross-validation test set features to each trained classifier. The best claimed scores of the individual classifiers are reported in Table 1 of [45] and exceed 0.94.

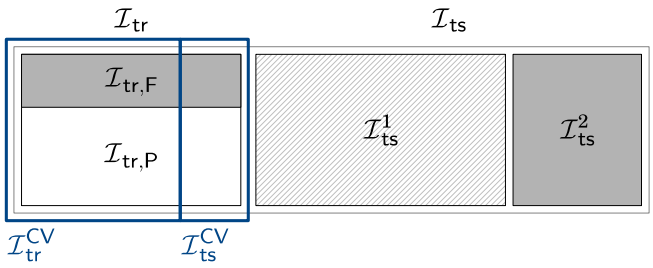


Figure 2.6: Cross-validation as performed in the Challenge training set.

Cozzolino, Gragnaniello, and Verdoliva, then, proceed with progressively concatenating the best submodels so to form a bigger submodel, and the whole training procedure is repeated. The concatenation continues until the length of the merged submodel exceeds the cardinality of the cross-validation training set. The claimed best Phase 1 score on $\mathcal{I}_{\text{tr}}$ is **0.9531**.

To further increase the score, PatchMatch is also computed on each image; since the false detection rate (i.e. pristine images being labeled as fake) of this method is extremely low (5 false matches over 1050 pristine images), the image is thus labeled as being fake whenever a copy-move is detected,

[3] See Appendix A

independently from the feature-based classifiers. Combining PatchMatch with the best reported achieved score on the training set, the score increases to 0.9737. As for the Challenge test set results, Cozzolino, Gragnaniello, and Verdoliva achieved the winning score of **0.9429**.

Localization

The promising approach developed by Team GRIP for the detection phase of the Challenge has been adapted with the purpose of detecting local tampers, along with a more conventional PRNU-based method. Here, the key idea is to merge three different masks produced by three different detectors: i) PatchMatch [23]; ii) local descriptors (i.e. the learned SVM on local features) with iii) PRNU. As these masks tackle different kinds of tampering, they need to be fused carefully, therefore a sequential strategy is developed.

Since the Challenge dataset is fully blind, no information is available on the cameras which have been used for taking the supplied snapshots, so the operations of camera attribution and fingerprint estimation are necessarily intertwined. Furthermore, a significant amount of images are needed for a successful PRNU extraction, therefore a clustering is attempted in order to be able to extract camera fingerprints $\widehat{K}$ from collections of noise residuals W.

The clustering algorithm is a variant of the *Pairwise Nearest Neighbour* (PNN), and alternates between hierarchically associating *closest* images and fingerprint estimation inside each identified cluster. The used distance measure is the PCE, with a threshold of 50; despite having been engineered for low resource usage, the task is still computationally demanding. After having identified the candidate clusters, camera fingerprints are recomputed and the camera attribution takes place, notwithstanding with a higher PCE threshold. Due to the blindness of the Challenge, the whole PRNU procedure is unreliable and needs to be taken with a grain of salt: we recall that, among the hypotheses, PRNU estimation forbids any scaling, heavy compression and misalignment between images used for estimating $\widehat{K}$.

The PatchMatch step is aimed at detecting copy-move forgeries; the algorithm quickly detects similar parts in an image which may indicate that a cloning has been performed. However, the biggest drawback of PatchMatch is that it does not distinguish the source region from the destination, therefore the obtained mask needs to be disambiguated in some way. Another drawback is that PatchMatch fails whenever the tampering is not a copy-move attack: as a consequence, a PatchMatch mask is not always available along the training dataset.

Finally, the aforementioned approach based on co-occurrence features has

been adapted to work in this situation. First, the images are partitioned into a grid of 128×128 pixel sliding blocks, where each one is labeled as "fake" if it is composed from 20% to 80% of forged pixels: the ansatz is to be able to detect forgeries by observing discontinuities on the edges of the modified region. Next, the said features are extracted and fed into a linear SVM classifier. Along with each predicted block label, one can also compute the distance of the feature vector from the hyperplane which separates the fake class from the pristine one. Such distance is used as a measure of confidence: the larger the distance, the more reliable the classification. This time, the obtained mask is not binary any more, but needs to be thresholded first.

With the three masks at hand, the fusion strategy prioritizes PatchMatch output (if available), eventually disambiguated using the PRNU map, if sufficiently reliable. Whenever PatchMatch fails to detect anything, PRNU map is used next, keeping local descriptors as a last resort. A chart summarizing the decision process for this strategy is represented in Figure 2.7.

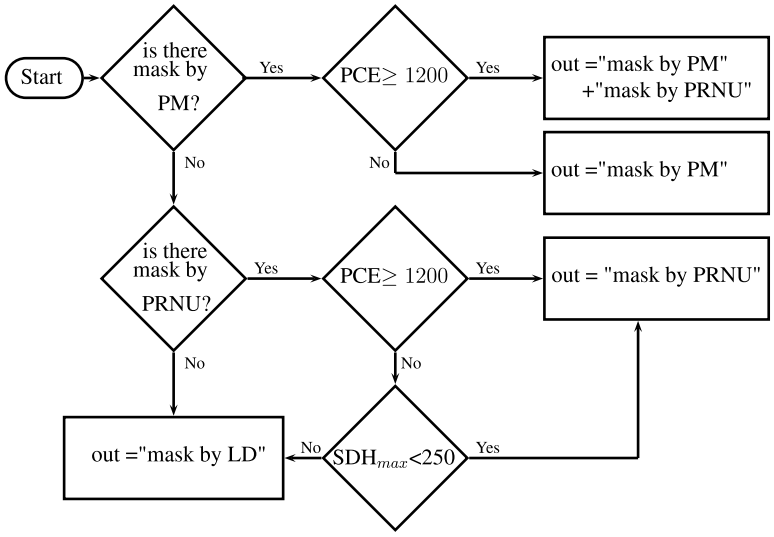


Figure 2.7: The fusion strategy flowchart. The feature-based approach is labeled as LD. *(Figure from [47])*

Regarding Challenge results, Cozzolino, Gragnaniello, and Verdoliva claimed to have reached a F_1 score in the training set of $\mathbf{0.4153}$, whereas the winning score achieved on the Challenge test set is $\mathbf{0.407172}$.

Chapter 3

Image tampering detection

In this chapter, we tackle the Detection problem paying particular attention to the Challenge Phase 1 dataset. We remind that the Detection problem is to be able to recognize whether a given image is fake or not. This problem is noticeably simpler than the Localization one, which will be approached in the next chapter: as a matter of fact, there are some tools which are able to distinguish statistics which can be hardly found in natural images, such as perfecly cloned objects or edges too smooth with respect to the rest of the image. To detect these features is sufficient to solve the Detection problem, but their exact localization is not always accurate.

We begin by implementing a variant of the winning approach described in Subsection 2.3.2: our method is built on the same battery of 39 feature-based detectors, as described in Subsection 2.3.2. The reason for this choice is that global features seemed to work well with the dataset, and they were exploited by the top two teams in the Challenge.

Once the feature detectors have been established, we analyze their general properties on the dataset. In this section we will also make use of the knowledge of the ground truth on the Challenge Phase 1 test set, by actually comparing the detector performance between training and test cases.

Next, we diverge from the winning submission by conceiving several means of producing a single decision from all 39 classifier outputs: we will also introduce a more advanced detector, built directly on the outputs of our 39 trained classifiers, showing that further improvement from the winning score is achievable.

Since the implemented detector is a technique heavily based on a machine learning approach, to actually understand the reason for its top-notch performance is a conditio sine qua non for its application in a more general context. As a consequence, we end this chapter with a targeted analysis of the Chal-

lenge dataset. A byproduct of the last step is a tentative adaptation of the algorithm for the Localization approach, which, however, will be carried out in the next chapter.

3.1 Proposed algorithm

3.1.1 Feature-based detectors

The feature-based detector is the same as the one which has been used in the winning submission (2.3.2). We represented it graphically in Figure 3.1.

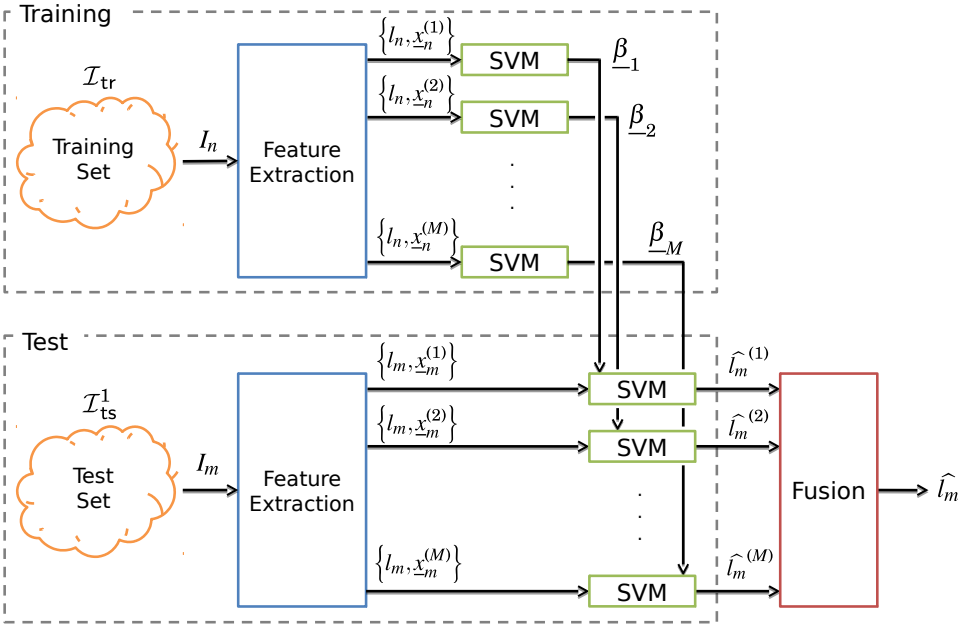


Figure 3.1: The SVM block diagram. For the sake of simplicity, cross-validation phase is not shown.

Briefly, to each image we associate a vector of 12753 features, grouped in $M = 39$ *submodels*, each one comprising either 325 or 338 features. Features in every submodel are positive values which sum to 1. To each image we also associate a ground truth label, which the classifiers will be trained on, in a supervised manner. Let $\underline{x}_i^{(m)}$ be the feature vector assigned to image $i \in \{1, \ldots, |\mathcal{I}|\}$ as extracted using the submodel $m \in \{1, \ldots, M\}$, and let $l_i \in \{0, 1\}$ be the true image label in binary form, where "1" indicates that i-th image is fake.

We will denote a prediction of l_i (e.g. as a classifier output) with $\widehat{l}_i$. Feature vectors in $\mathcal{I}_{tr}$ can be organized in a matrix form: the resulting matrix has 1500 rows, one per image, and $6 \times 338 + 33 \times 325 = 12753$ columns.

As in Subsection 2.3.2, the training set $\mathcal{I}_{\text{tr}}$ is randomly split into $\mathcal{I}_{\text{tr}}^{\text{CV}}$ and $\mathcal{I}_{\text{ts}}^{\text{CV}}$ (see Figure 2.6). Here, we report our choice for cross-validation setup:

$$\begin{aligned} \mathcal{I}_{\text{tr}} &= \mathcal{I}_{\text{tr}}^{\text{CV}} \cup \mathcal{I}_{\text{ts}}^{\text{CV}} \\ \left|\mathcal{I}_{\text{tr}}^{\text{CV}} \cap \mathcal{I}_{\text{tr,F}}\right| &= {}^{5}\!/_{6} \left|\mathcal{I}_{\text{tr,F}}\right| \\ \left|\mathcal{I}_{\text{tr}}^{\text{CV}} \cap \mathcal{I}_{\text{tr,P}}\right| &= {}^{5}\!/_{6} \left|\mathcal{I}_{\text{tr,P}}\right| \end{aligned}$$

For each submodel m, a SVM binary classifier $\underline{\overline{\beta}}^{(m)}$ is trained on the feature set $\left\{\underline{x}_i^{(m)}\right\}_{i \in \mathcal{I}_{\text{tr}}^{\text{CV}}}$, using the image labels $\left\{l_i\right\}_{i \in \mathcal{I}_{\text{tr}}^{\text{CV}}}$ as supervised inputs.

To be effective for SVM classification, feature space needs to be transformed first: we experimentally found out that matrix standardization is the best choice on this dataset.

Secondly, SVMs require calibration, as a number of parameters (here, *metaparameters*) have to be specified: to this purpose, we perform 5-fold cross-validation inside $\mathcal{I}_{\text{tr}}^{\text{CV}}$ and we chose metaparameters which maximize the mean Phase 1 score, computed using the predicted outputs $\left\{\widehat{l}_i^{(m)}\right\}_{i \in \mathcal{I}_{\text{tr}}^{\text{CV}}}$. The strategy for SVM metaparameter selection is a simple grid search on a very coarse grid, further refined if necessary.

Next, M SVM classifiers $\underline{\beta}^{(m)}$ are trained on the whole $\left\{\underline{x}_i^{(m)}\right\}_{i \in \mathcal{I}_{\text{tr}}}$ along with $\left\{l_i\right\}_{i \in \mathcal{I}_{\text{tr}}}$, using the learned best metaparameters. Each $\underline{\beta}^{(m)}$, once trained, is then used to predict the test set labels $\left\{\widehat{l}_i^{(m)}\right\}_{i \in \mathcal{I}_{\text{ts}}^{1}}$, whose Challenge results will be computed onto.

As our SVM works on linear bases, to each image i we also associate the soft decision values $d_i^{(m)}$, where

$$\widehat{l}_i^{(m)} = \frac{1}{2}\left(\text{sign}(d_i^{(m)}) + 1\right) \tag{3.1}$$

3.1.2 Simple ensemble classifiers

Up to now, for each image we have $39 \times 2 = 78$ different outputs: the predicted labels and the decision values. Our goal is, then, to obtain a single decision from the sets $\left\{\widehat{l}_i^{(m)}\right\}_{m=1}^{K}$ and $\left\{d_i^{(m)}\right\}_{m=1}^{K}$. A way to choose such strategy is to implement an ensemble classifier.

Here, we distinguish what we call *simple* ensemble classifiers (i.e. which require no training) from more advanced ones, such as the *boosting classifier*.

Definition 1 (Simple ensemble classifier). *A* simple ensemble classifier *is a function*

$$f\colon \{0,1\}^{39} \times \mathbb{R}^{39} \to \{0,1\} \qquad (3.2)$$
$$\{\widehat{l}_i^{(m)}\}_{m=1}^{39} \times \{d_i^{(m)}\}_{m=1}^{39} \mapsto \widehat{l}_i$$

For sake of simplicity, let us drop the image index i, and collect each submodel output in a vector:

$$\underline{l} := \left\{\widehat{l}_i^{(m)}\right\}_{m=1}^{39} \qquad \underline{d} := \left\{d_i^{(m)}\right\}_{m=1}^{39}$$

We considered the following set of simple ensemble classifiers:

- *Simple Majority*

$$f_{\mathrm{SM}}(\underline{l},\underline{d}) := \begin{cases} 1 & \text{if} \quad \frac{1}{39}\sum_{m=1}^{39}\underline{l}_m > \frac{1}{2} \\ 0 & \text{else} \end{cases}$$

- *Signed Sum*

$$f_{\mathrm{SS}}(\underline{l},\underline{d}) := \begin{cases} 1 & \text{if} \quad \sum_{m=1}^{39}\underline{d}_m > 0 \\ 0 & \text{else} \end{cases}$$

- *Weighted Signed Sum*

$$f_{\mathrm{WSS}}(\underline{l},\underline{d}) := \begin{cases} 1 & \text{if} \quad \sum_{m=1}^{39}\left(2\Phi(\underline{d}_m)-1\right) > 0 \\ 0 & \text{else} \end{cases}$$

  where $\Phi(x) := \dfrac{1}{1+e^{-x}}$ is the sigmoid function.

- *Signed* L^2

$$f_{\mathrm{L}^2}(\underline{l},\underline{d}) := \begin{cases} 1 & \text{if} \quad \sum_{m=1}^{39}\left|\underline{d}_m\right|^2\underline{l}_m > 0 \\ 0 & \text{else} \end{cases}$$

We found out (see 3.2.5) that majority voting is very simple, requires no training and enables us to almost achieve the claimed Phase 1 score with no effort.

3.1.3 Best k submodels

A variant on the previous approach is to consider only the k highest scoring submodels on the training set. Results are available in Figure 3.7: in most cases, the improvement is significant when choosing less than all 39 submodels.

3.1.4 Boosting

In addition to simple ensemble classifiers, as implemented in the previous subsection, we also briefly investigated a more advanced approach built directly on the output of all 39 classifiers.

The idea is to train a device which learns from examples which are misclassified by a set of *weak learners*: the obtained classifier is stronger than the ensemble (hence, the name *strong learner*), although an additional training step is required, thus necessitating more data. This procedure is named *boosting*. To implement a boosting classifier, we performed a logistic regression with the decision values of each SVM as 39 predictor variables, and the image binary label l_i as the outcome variable.

In equations, the model is written as

$$\operatorname{logit}\left(\mathbb{E}\left[\, l_i \,\middle|\, d_i^{(1)}, \dots, d_i^{(39)} \,\right]\right) = b_0 + \sum_{m=1}^{39} b_i d_i^{(m)} \qquad \forall i \in \mathcal{I}_{\mathsf{tr}}\,, \tag{3.3}$$

where $\operatorname{logit}(x) \coloneqq \log \frac{x}{1-x}$ is the logistic function, $d_i^{(m)}$ is the decision value for the i-th image as returned by the m-th classifier, and $\{b_i\}_{i=0}^{39}$ are the unknown coefficients of the boosting classifier. Once all the $\{b_i\}_{i=0}^{39}$ are obtained, the classifier can be used to predict $\widehat{l_i}$ on the Challenge test set $\mathcal{I}_{\mathsf{ts}}^1$.

Due to the lack of time and large computational effort, no dimensional reduction is attempted, relying instead on the proven capabilities of linear SVMs to work reliably on very high dimensional data [48]. A much more deep analysis would be required, starting perhaps from Linear Discriminant Analysis (as in [46]) or its kernel-based variant. More remarks on dimensional reduction can be found in 3.2.4.

3.2 Experimental results

3.2.1 Problem setup

We start by summarizing our data in Table 3.1. To perform Support Vector Machine classification, we used `LIBLINEAR` [49], a free open-source library built specifically for large datasets. Citing from the accompanying guide [49], `LIBLINEAR` implements L^1-loss and L^2-loss binary and multi-class SVMs with linear basis along with primal and dual solvers and L^1 or L^2 regularization: it also supports probability outputs through L^2-regularized logistic regression. In our problem, we chose L^2-loss along with L^2 regularization; prior class probabilities are always set to the sample class proportions.

First, a uniform cross-validation set $\mathcal{I}_{\mathsf{tr}}^{\mathsf{CV}}$ is randomly extracted from $\mathcal{I}_{\mathsf{tr}}$ with the aforementioned proportions. For now, let's choose submodel m. We

Data	Amount
Submodel length	328 or 325
Number of submodels	39
Total number of features	12753
$\lvert\mathcal{I}_{\mathsf{tr}}\rvert$	1500
$\lvert\mathcal{I}_{\mathsf{tr}}^{\mathsf{CV}}\rvert$	1250
$\lvert\mathcal{I}_{\mathsf{ts}}^{\mathsf{CV}}\rvert$	250
$\lvert\mathcal{I}_{\mathsf{ts}}^{1}\rvert$	5713.

Table 3.1: Detection dataset summary.

can now build the full submodel data matrix $X^{(m)}$ by stacking our feature column vectors $\underline{x}_i^{(m)}$:

$$X^{(m)} = \begin{pmatrix} \underline{x}_1^{(m)\top} \\ \underline{x}_2^{(m)\top} \\ \vdots \\ \underline{x}_N^{(m)\top} \end{pmatrix} \in \mathbb{R}^{N\times p} \tag{3.4}$$

where $N = |\mathcal{I}_{\mathsf{tr}}|$ and $p \in \{325, 338\}$ is the number of features expressed by submodel m. To enhance the score, we chose to standardize our features: specifically, we standardized each column, considering only entries whose rows are in $\mathcal{I}_{\mathsf{tr}}^{\mathsf{CV}}$, and we applied the obtained transformation to the cross-validation test set $\mathcal{I}_{\mathsf{ts}}^{\mathsf{CV}}$.

3.2.2 SVM calibration

Let $\underline{\beta}^{(m)} \in \mathbb{R}^p$ be the coefficients assigned to the m-th classifier. As a slight abuse of notation, with $\underline{\beta}^{(m)}$ we will both refer to the SVM on submodel m and the coefficients of the same SVM. From the model in [49], once trained on $\mathcal{I}_{\mathsf{tr}}^{\mathsf{CV}}$ each $\underline{\beta}^{(m)}$ solves

$$\underline{\beta}^{(m)} = \underset{\underline{w}\in\mathbb{R}^p}{\arg\min} \left\{ \frac{1}{2}\underline{w}^\top \underline{w} + C^{(m)} \sum_{i\in\mathcal{I}_{\mathsf{tr}}^{\mathsf{CV}}} \max\left[0, 1-(2l_i-1)\,\underline{w}^\top \underline{x}_i^{(m)}\right] \right\}, \tag{3.5}$$

where $C^{(m)} > 0$ is a penalty parameter and the soft decision value is defined as

$$d_i^{(m)} := \underline{w}^\top \underline{x}_i^{(m)} \tag{3.6}$$

As said before, calibration of $C^{(m)}$ is done for each submodel m using an exhaustive search on 11 values logarithmically equispaced inside $\{2^{-20}, 2^{20}\}$. To

increase robustness, we try to avoid overfitting by performing 5-fold cross-validation 5 times inside $\mathcal{I}_{\mathsf{tr}}^{\mathsf{CV}}$: the final score for each choice of $C^{(m)}$ is then set to be the mean of the scores on the remaining folds. Finally, the grid is refined once around the best choice of $C^{(m)}$. The calibration process is shown in Figure 3.2. The dependence of the score on the choice of $C^{(m)}$ is quite flat (Figure 3.2a): moreover, the score increment on each best parameter before and after refinement (Figure 3.2b) shows that grid refinement is not necessary.

Several experiments have been performed by choosing a radial basis SVM: obtained scores were comparable to linear basis SVM, but required calibrating two metaparameters (hence with a quadratic complexity), with a significantly longer training time per metaparameter choice.

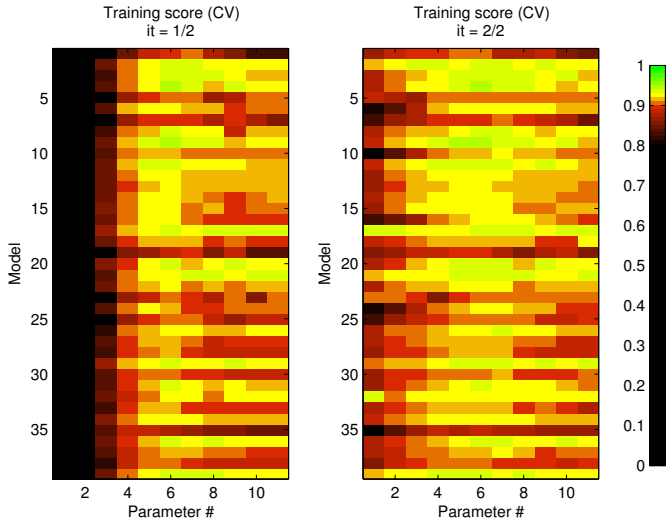
3.2.3 Performance on each test set

After calibration, each SVM is tested on the cross-validation test set $\mathcal{I}_{\mathsf{ts}}^{\mathsf{CV}}$, thus producing a set of predicted labels $\left\{\widehat{l}_i^{(m)}\right\}_{i\in\mathcal{I}_{\mathsf{ts}}^{\mathsf{CV}}}$ along with the respective scores. In Figure 3.3 we show through repetition of the cross-validation procedure (i.e. 100 random choices of $[\mathcal{I}_{\mathsf{tr}}^{\mathsf{CV}}, \mathcal{I}_{\mathsf{ts}}^{\mathsf{CV}}]$) that the Challenge training set $\mathcal{I}_{\mathsf{tr}}$ is uniform in score for classification purposes.

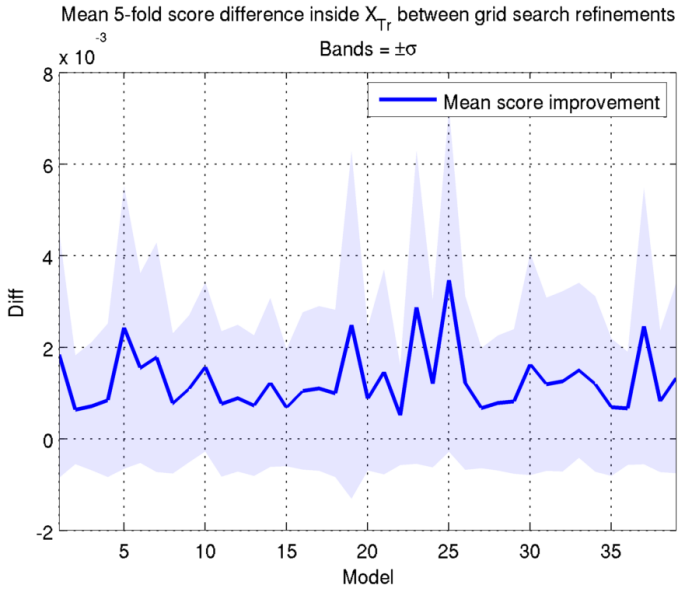
As the ground truth labels in the Challenge Phase 1 test set are available, we then proceed to evaluate the performance of each $\underline{\beta}^{(m)}$ on $\mathcal{I}_{\mathsf{ts}}^{1}$. First, we begin by retraining each $\underline{\beta}^{(m)}$ on the entire training set $\mathcal{I}_{\mathsf{tr}}$, using the best metaparameters learned on $\mathcal{I}_{\mathsf{tr}}^{\mathsf{CV}}$. Next, to each SVM we feed $\mathcal{I}_{\mathsf{ts}}^{1}$ to obtain predicted labels, decision values and the Challenge Phase 1 score.

From Figure 3.4 we immediately observe a loss in score, although it still remains very high. The loss is not due to a particular choice of the cross-validation set, and it persists even if we change each one of the metaparameters $C^{(m)}$ to make them span over the grid used in calibration (see Figure 3.4b). We conjecture that the reason behind it is a structural difference between the Challenge training set and the Challenge test set.

To further investigate such loss, we can exploit an *oracle* by training our SVMs on $\mathcal{I}_{\mathsf{tr}}$ while seeking for the best $\{C^{(m)}\}_{m=1}^{39}$ that maximizes the score on $\mathcal{I}_{\mathsf{ts}}^{1}$. As Figure 3.5 shows, we cannot hope to reach a much better score. Moreover, from Figure 3.5c we also note that scores on $\mathcal{I}_{\mathsf{ts}}^{1}$ are more stable than scores obtained against $\mathcal{I}_{\mathsf{ts}}^{\mathsf{CV}}$: this is against all odds since the Challenge test set is much larger than the cross-validation test set ($\left|\mathcal{I}_{\mathsf{ts}}^{1}\right| = 5713$, while $\left|\mathcal{I}_{\mathsf{ts}}^{\mathsf{CV}}\right| = 250$).



(a) Color: score, Y axis: submodel m, Panels: refinements, X axis: index of the chosen metaparameter inside m-th metaparameter grid



(b) Difference between maximum scores per submodel between refinements.

Figure 3.2: Metaparameter grid search. Each reported score is obtained by training on ⅘ of $\mathcal{I}_{\text{tr}}^{\text{CV}}$ and testing on the remaining ⅕: this procedure is repeated 5 times, and the resulting scores are averaged. Note how quite insensitive the score is on the choice of $C^{(m)}$: we remind that in the first plot of (a), the x-axis ranges inside $\{2^{-20}, 2^{20}\}$. In (b) we show the negligible score increments for each submodel between the first and the second grid refinement.

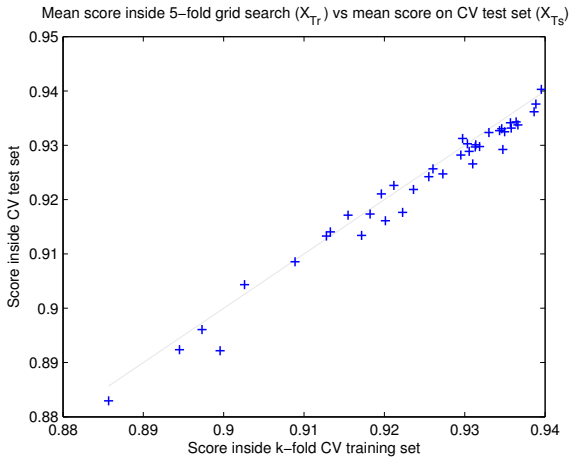


Figure 3.3: Mean 5-fold cross-validation score inside $\mathcal{I}_{\text{tr}}^{\text{CV}}$ *for each submodel (+) against mean score on* $\mathcal{I}_{\text{ts}}^{\text{CV}}$*. Means over* 100 *realizations of* $\left[\mathcal{I}_{\text{tr}}^{\text{CV}}, \mathcal{I}_{\text{ts}}^{\text{CV}}\right]$*. Gray line: bisector.*

3.2.4 On dimensional reduction

We would like to remark that no dimensional reduction has been performed: as a result, the observed loss in score may be simply due to feature dimensionality and the different sample sizes, thus a much longer analysis would be required. Still, we cite the following result by Cover [50]:

Theorem 1. *Let* $X = \left\{\underline{x}_i^{(m)}\right\}_{i\in\mathcal{I}}$ *be a set of* $N = |\mathcal{I}|$ *p-dimensional features chosen, both randomly and independently, on* $\mathbb{R}^p$ *through a probability measure* μ *on* $\mathbb{R}^p$*. Suppose that every subset* $A \in 2^X : |A| = p$ *is linearly independent. Now, let* A, B *be sets such that* $A \subset X$*,* $B = X \setminus A$*. Then, the probability* $P(N,p)$ *that* A *is* linearly separable *from* B *is:*

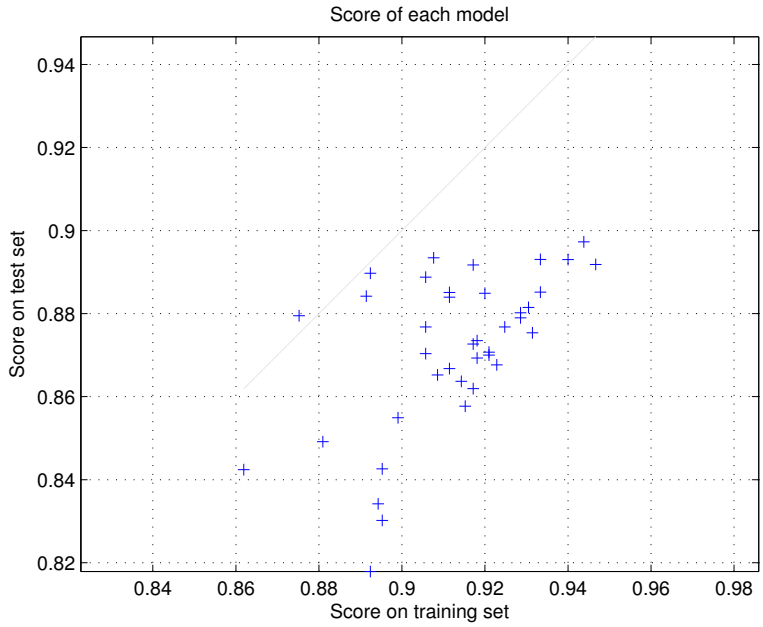
$$P(N,p) = \left(\frac{1}{2}\right)^{N-1} \sum_{k=0}^{p-1} \binom{N-1}{k} \tag{3.7}$$

Let us consider the training case where $X = \left\{\underline{x}_i^{(m)}\right\}_{i\in\mathcal{I}_{\text{tr}}}$, p is 338, and let $\{A,B\}$ be the partition of X as identified by the SVM. We have $N = |\mathcal{I}_{\text{tr}}| = 1500$, $p = 338$. Plugging those numbers in Equation (3.7), the probability that the partitioning is solely due to chance is:

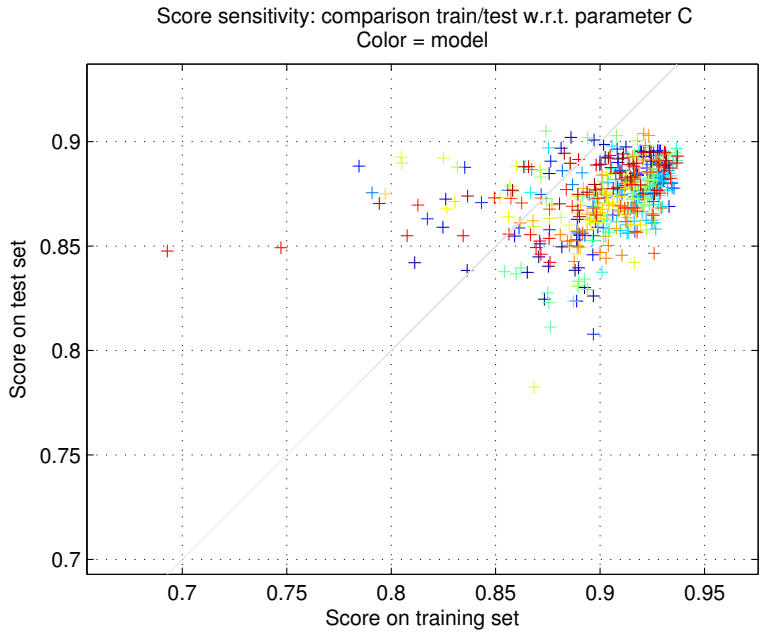
$$P(1500, 338) = 1.7410^{-106}$$

while to reach the common p-value of 0.05, a sample size of at least 720 images is required.

We note, however, that this result does not exploit the cross-validation mechanism: in fact, we conjecture that the error committed due to high data

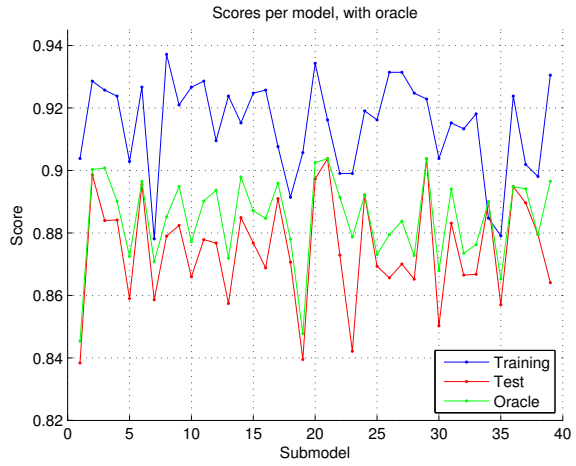


(a) Score on $\mathcal{I}_{\text{ts}}^{\text{CV}}$ by training on $\mathcal{I}_{\text{tr}}^{\text{CV}}$ against score on $\mathcal{I}_{\text{ts}}^{1}$ by training on $\mathcal{I}_{\text{tr}}$

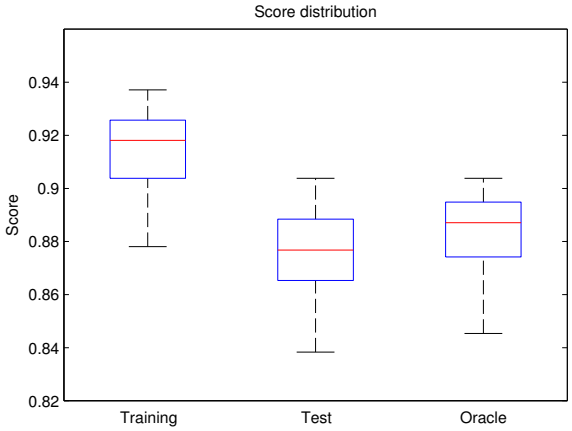


(b) As (a), but varying $C^{(m)}$ 11 times around the best value on $\mathcal{I}_{\text{ts}}^{\text{CV}}$, grouped by submodel (colors)

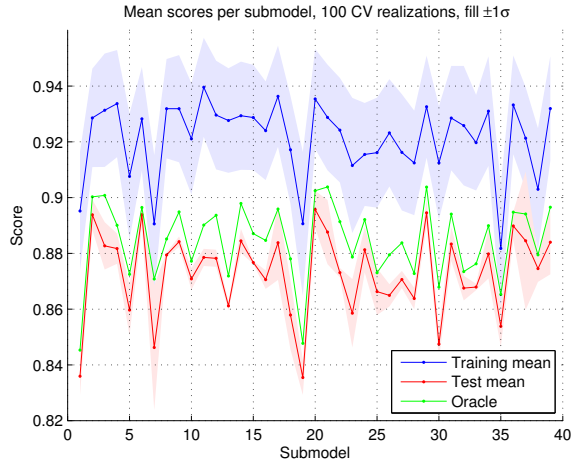
Figure 3.4: Comparison between score in $\mathcal{I}_{\text{tr}}$ and $\mathcal{I}_{\text{ts}}^{1}$. Figure (a) shows that each submodel performs worse if it is trained on $\mathcal{I}_{\text{tr}}$ and it is tested on $\mathcal{I}_{\text{ts}}^{1}$, rather than having been trained on $\mathcal{I}_{\text{tr}}^{\text{CV}}$ and tested on $\mathcal{I}_{\text{ts}}^{\text{CV}}$. (compare with Figure 3.3).
Plot (b) shows that this loss does not depend on the choice of $C^{(m)}$: we swept each parameter on the range used in the last grid search iteration, but reported scores in the Phase 1 test set were almost always lower than scores on the cross-validationtest set. This hints for some underlying difference between the two datasets, which is not expressed inside $\mathcal{I}_{\text{tr}}$.



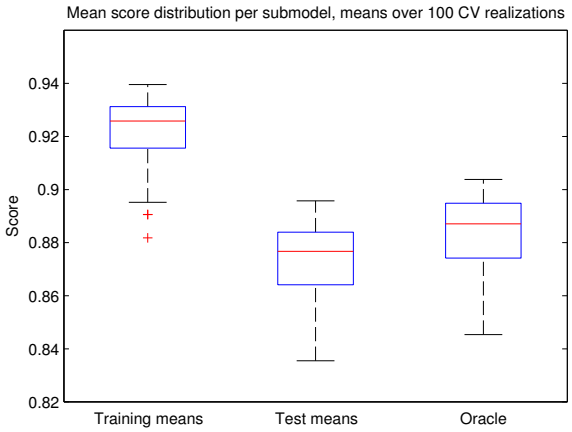
(a) Score distribution across submodels



(b) Score distribution across submodels, boxplots



(c) Mean score distribution along submodels, means across 100 cross-validation realizations, the fill extends up to a standard deviation from the mean.



(d) Distribution of mean score along submodels, 100 cross-validation realizations.

Figure 3.5: Usage of an oracle on $\mathcal{I}^1_{\text{ts}}$. We compare SVMs trained on $\mathcal{I}^{\text{CV}}_{\text{tr}}$ and tested on $\mathcal{I}^{\text{CV}}_{\text{ts}}$ against SVMs trained on $\mathcal{I}_{\text{tr}}$ and tested on $\mathcal{I}^1_{\text{ts}}$. In the second row, we repeat the same experiment 100 times, each one resampling $\mathcal{I}^{\text{CV}}_{\text{tr}}$ and $\mathcal{I}^{\text{CV}}_{\text{ts}}$.
The oracle, instead, is trained on $\mathcal{I}_{\text{tr}}$ and tested on $\mathcal{I}^1_{\text{ts}}$: however, metaparameters $C^{(m)}$ are chosen in order to maximize its score on $\mathcal{I}^1_{\text{ts}}$: this gives us an upper bound on each classifier score. It is clear how we cannot expect our classifiers to perform equally well on the test set rather than the training set.

dimensionality is (partially) controlled by the usage of the test set. Also, the framework of Theorem 1 is applicable for "fully random" classifiers: in practice, the explored sample space (that is, the possible set dichotomies) is much smaller and distorted due to SVM penalization and metaparameter selection. Thus, it would be more helpful to characterize a "random" to classifier to compare against, numerically or analytically, perhaps starting with an unoptimized linear classifier. This procedure would also clarify us the role of high dimensionality on the Challenge score.

Again, a much more thorough analysis would be mandatory to assess hypotheses in Theorem 1. From now on, we will take for granted our results without approving (dismissing) them on the grounds of statistical (non—) significance due to high dimensionality.

3.2.5 Ensemble classifiers

We proceed now with evaluating the performance of some ensemble classifiers. In Figure 3.6 we show the results of each simple ensemble classifier. No method is preferrable to any other, although any ensemble classifier gives a sizable improvement over the average decision taken by all classifiers.

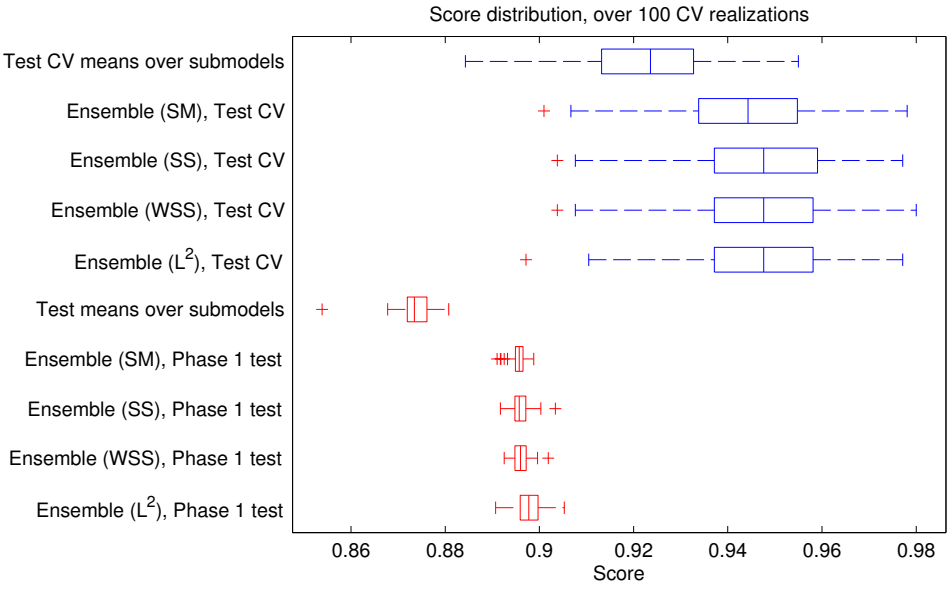


Figure 3.6: Simple ensemble classifier scores across 100 realizations of the cross-validation set. Blue boxplots: scores obtained by testing on $\mathcal{I}_{\mathsf{ts}}^{\mathsf{CV}}$, red boxplots: scores obtained by testing on $\mathcal{I}_{\mathsf{ts}}^{1}$. The first boxplot of each group is the cross-validation distribution of the sample mean of the scores across submodels.
Notice the sizable score gain obtained by choosing any *simple ensemble classifier.*

3.2.6 Best-k models

We can also modify the previous procedure by computing the score using only the k highest scoring submodels on each training set: this strategy is advantageous when k approaches 10 (Figure 3.7).

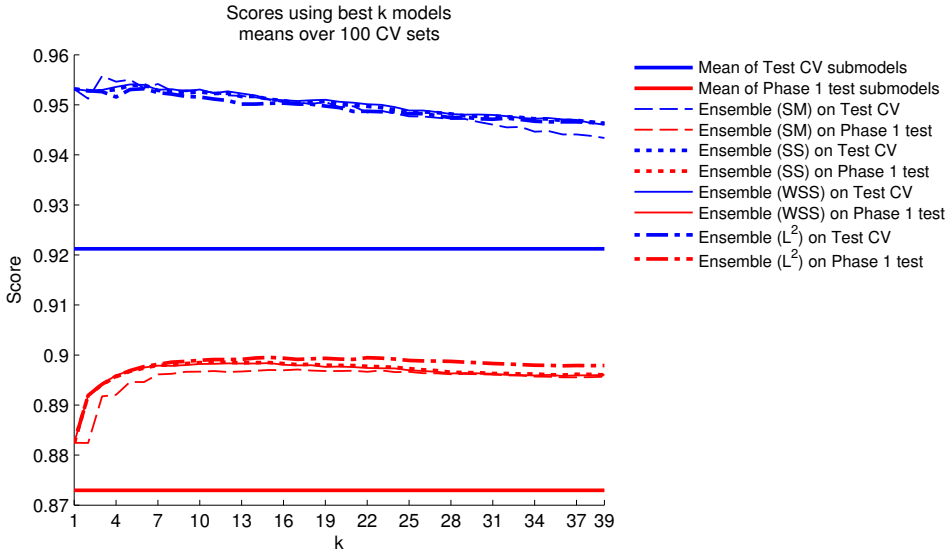


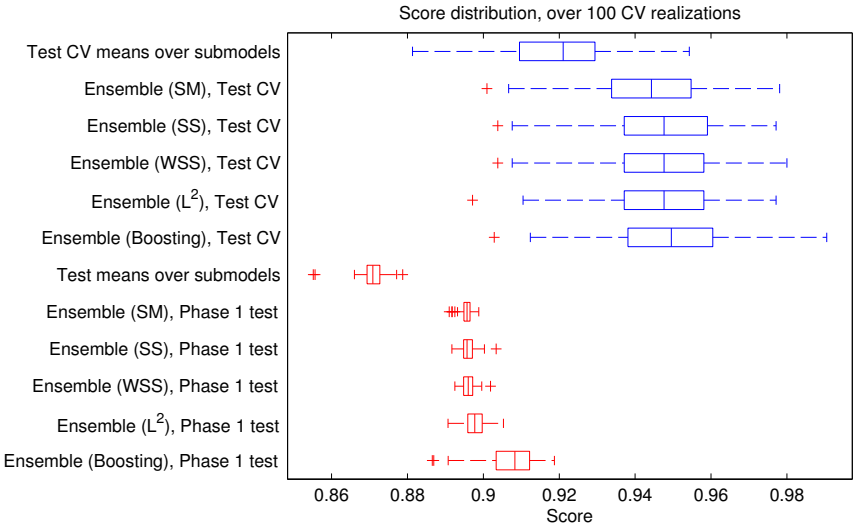
Figure 3.7: Simple ensemble classifier mean scores using best k submodels. Means across 100 cross-validation realizations.
Notice how deciding using the best performing classifier on $\mathcal{I}_{\text{tr}}$ leads us to a loss in score, although still being better than the average submodel.

3.2.7 Boosting

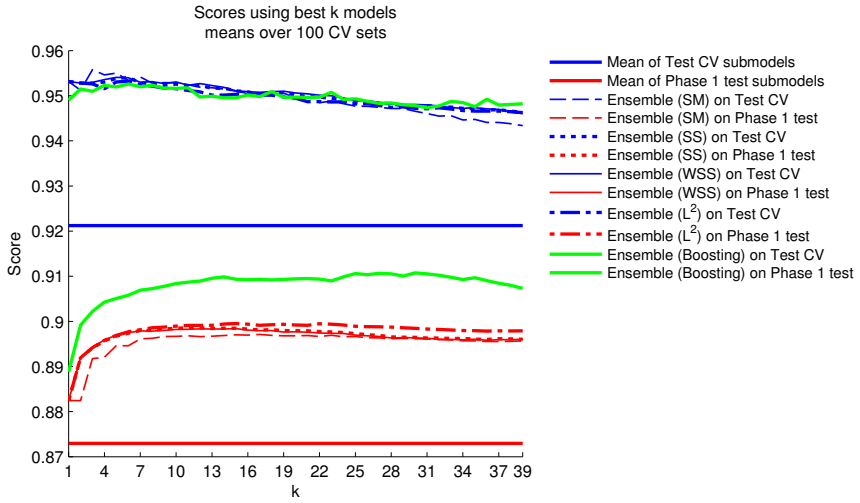
In principle, to train a boosting classifier, one ought to perform another cross-validation step: due to the lack of time, we did not implement this necessary precaution, training instead on $\mathcal{I}_{\text{tr}}^{\text{CV}}$ ($\mathcal{I}_{\text{tr}}$) and testing on $\mathcal{I}_{\text{ts}}^{\text{CV}}$ ($\mathcal{I}_{\text{ts}}^{1}$). As a result, obtained inferences have to be taken with a grain of salt. Moreover, perfect separation in the training set is achieved using $\left|\mathcal{I}_{\text{ts}}^{\text{CV}}\right|$ observations: that is, there exists a linear combination of 39 columns which perfecly predicts l_i from $\left\{d_i^{(m)}\right\}_{m=1}^{39}$ for all images $i \in \mathcal{I}_{\text{tr}}^{\text{CV}}$.

In statistical literature, this is deemed to be a pathological situation, requiring much more analysis in order to uncover the reason for the occurrence of the perfect separation; another tentative remedy is to apply some form of regularization. Nevertheless, we decided to show the results anyway, as a boosting approach could be potentially rewarding to the purpose of decision

fusion. An example plot is reported in Figure 3.8: from the plot, we appreciate that a sizable gain on test set is achieved.



(a) Same plot as Figure 3.6, with the boosting classifier added.



(b) Same plot as Figure 3.7, with the boosting classifier added (in green).

Figure 3.8: Boosting classifier performance.

3.3 Deeper analysis

We devised a way to distinguish fake images from pristine ones using features automatically learned from images. However, being an approach based on machine learning tools, the biggest drawback is that both features $\underline{x}_i^{(m)}$ and individual classifiers $\underline{\beta}^{(m)}$ lack a clear meaning: the whole analysis is, thus, incomplete. We therefore proceed to analyze their performance on custom-built datasets in order to understand what is actually learned from the input features.

3.3.1 Global classifiers on blocks

So far, we have built classifiers which perform very well both on the training set and on the test set, albeit with a small score penalty. It is natural to ask whether such effectiveness holds also on local domains. To this purpose we tested our previously trained $\{\underline{\beta}^{(m)}\}_{m=1}^{39}$ on non-overlapping blocks extracted from training images, using, e.g. the Weighted Signed Sum to decide whether a block is fake or not.

Most of the times, results are surprising as the analyzed classifiers are not able to recognize anything, often taking completely wrong decisions: an example is shown in Figure 3.10. This behaviour is also mostly independent from the size of each block, conditioned on being large enough. Notice that in some cases we can also obtain a decision closer to the truth (see Figure 3.9): however, this happens very rarely.

An extreme example of this anomaly can be found in Figure 3.11: this also shows that the output of each block classifier is clearly biased by the label assigned to the image where the blocks came from. In fact, it is reasonable to assume that, due to the very large block size, block and image features are approximately equal, therefore each classifiers performs as if it were analyzing the entire (fake) image.

3.3.2 New datasets

From this set of remarks, we conjecture that these classifiers are *not* able to distinguish modified images from pristine ones based on local features. To further establish this ansatz, we proceeded to create a series of four scenarios by sampling from the one pertaining to the Challenge. Afterwards, the classifier presented in Section 3.1 was trained and tested according to each one of them. We depicted these scenarios in Figure 3.12.

- **Scenario A**: $\mathcal{I}_{\mathsf{tr}}^{\mathsf{A}}$
  In this scenario, the training set $\mathcal{I}_{\mathsf{tr}}^{\mathsf{A}} = \mathcal{I}_{\mathsf{tr}}$ was composed by the whole

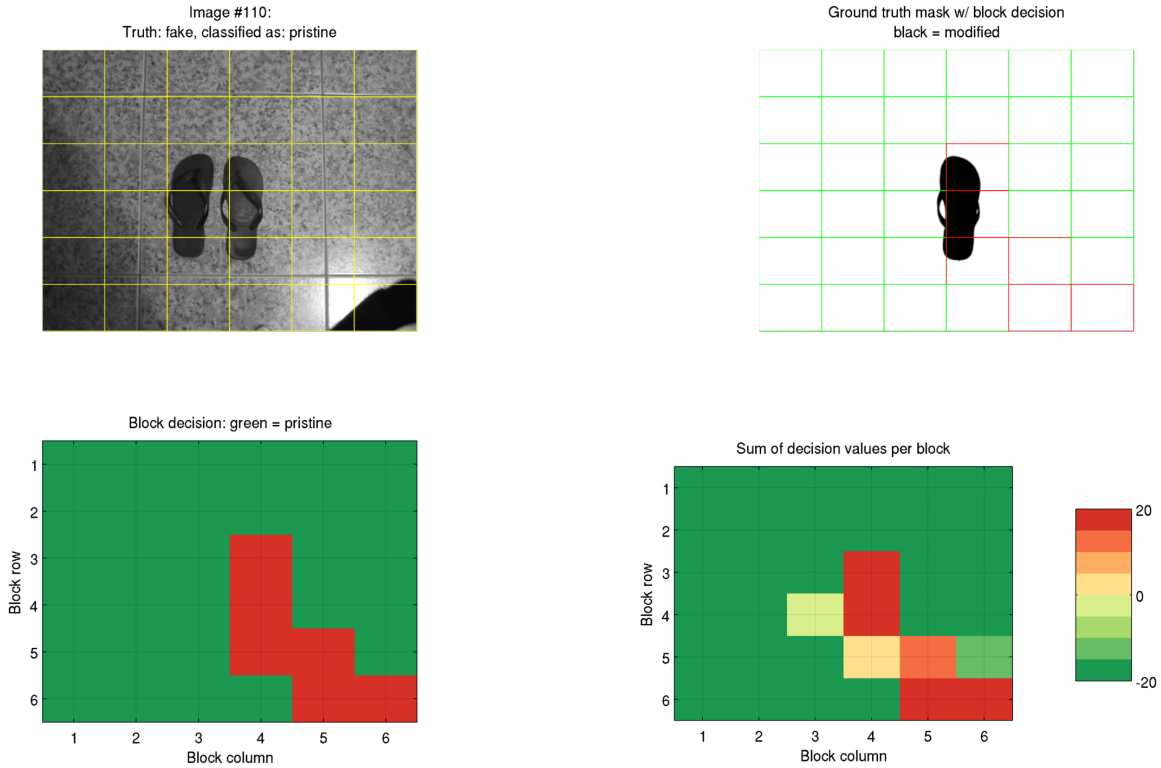


Figure 3.9: Results of applying the global procedure on blocks extracted from images. In this case, this method correctly mostly distinguishes the tampered region from the rest of the image, as the ground truth masks shows (top right plot).

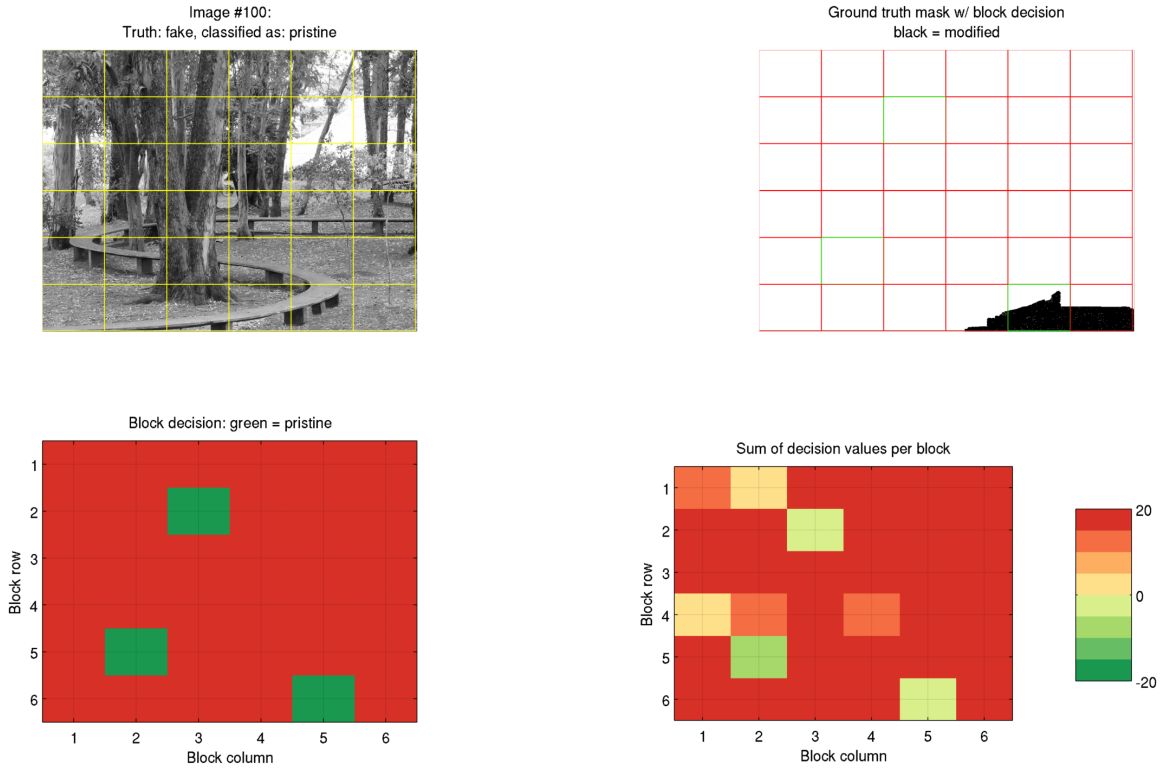


Figure 3.10: Results of applying the global procedure on blocks extracted from images. In this case, this method classifies the images as being almost fully modified, whereas the image has been only tampered in its lower right portion.

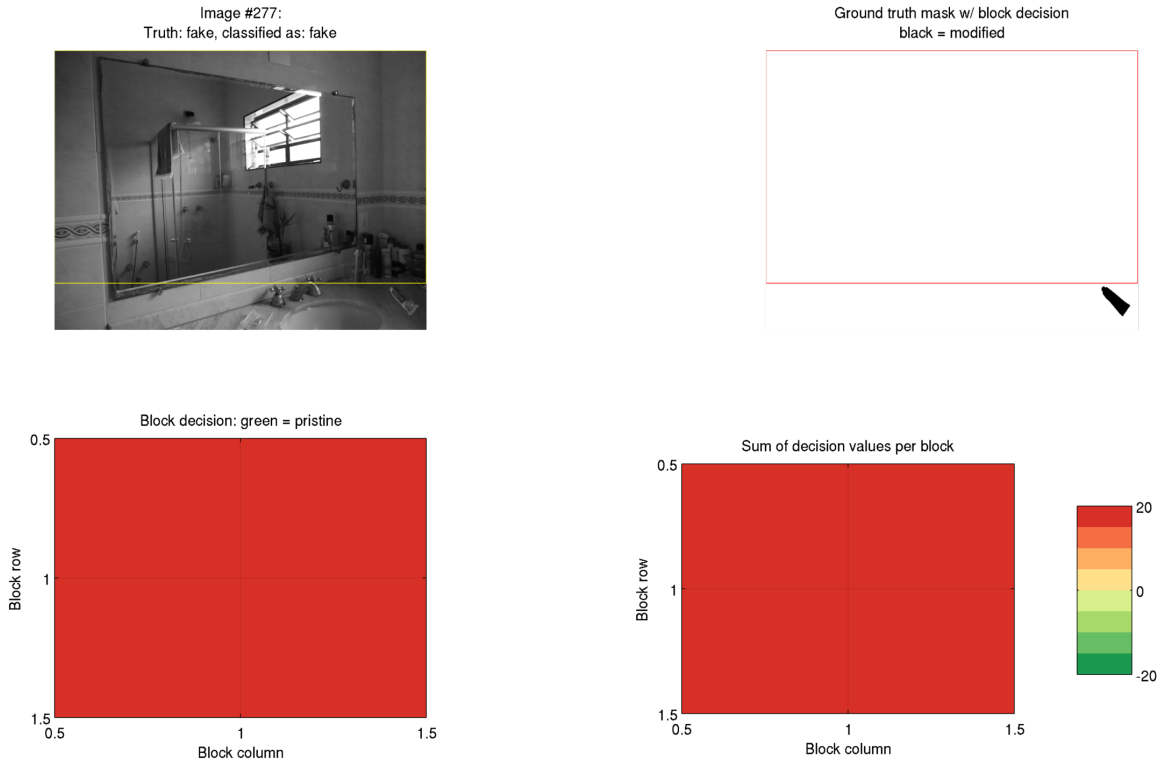


Figure 3.11: Results of applying the global procedure on blocks extracted from images. In this case, we extracted a single *block which almost comprises the entire image except for a small part, where the tampering is located.*

Phase 1 training set. The class label $l_i = 0$ was associated to *pristine* images, while $l_i = 1$ to *fake* ones (see Figure 3.12a).

In this scenario we expect that the trained classifier performs well on the testing dataset.

- **Scenario B**: $\mathcal{I}_{\text{tr}}^{\text{B}}$
  In this scenario, we used as training set $\mathcal{I}_{\text{tr}}^{\text{B}}$ a set of 400×400 pixels non-overlapping blocks, taken from either *fake* or *pristine* images. More specifically, we associated $l_i = 0$ to blocks taken from *pristine* images, while $l_i = 1$ to blocks coming from *fake* images and containing part of the tampered area (see Figure 3.12b).

  This scenario is similar to Scenario A, despite the fact that we train the algorithm on smaller blocks rather than full images: therefore, we expect the algorithm to work properly in detecting the tamperings.

- **Scenario C**: $\mathcal{I}_{\text{tr}}^{\text{C}}$
  In this scenario, similarly to the previous one, the training set $\mathcal{I}_{\text{tr}}^{\text{C}}$ was composed selecting 400×400 pixels non-overlapping blocks. This time we only considered *fake* images of the Phase 1 training set. In particular we associated $l_i = 0$ to blocks not containing any altered pixel, and $l_i = 1$ to blocks containing at least part of the tampered region (see Figure 3.12c).

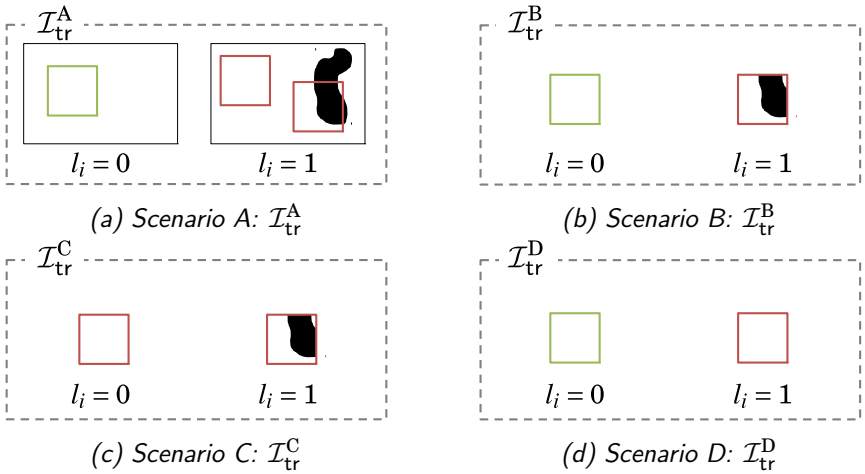
  This test aims to show whether the detection algorithm is capable to distinguish between tampered and non-tampered regions in the same image. Also in this case we would expect the algorithm to perform well.

- **Scenario D**: $\mathcal{I}_{\text{tr}}^{\text{D}}$
  Finally, in the last scenario, we used again as training set $\mathcal{I}_{\text{tr}}^{\text{D}}$ a set of 400×400 pixels non-overlapping blocks from the Phase 1 training set images $\mathcal{I}_{\text{tr}}$. In this case, we associated $l_i = 0$ to blocks coming from *pristine* images, and $l_i = 1$ to blocks coming from *fake* images not containing any forged pixel (see Figure 3.12d).

  In this scenario we expect the algorithm to fail, since all the blocks that we use do not contain any tampering.

Next, after training our classifiers on each training set scenario, we computed the scores using three different test sets. Obtained scores are summarized in Figure 3.13.

- *Cross-validation set*: Figure 3.13a shows the scores achieved testing each SVM on the dataset used to train it (i.e., the SVMs trained on $\mathcal{I}_{\text{tr}}^{\text{k}}$ were then tested on $\mathcal{I}_{\text{ts}}^{\text{k}} = \mathcal{I}_{\text{tr}}^{\text{k}}$).



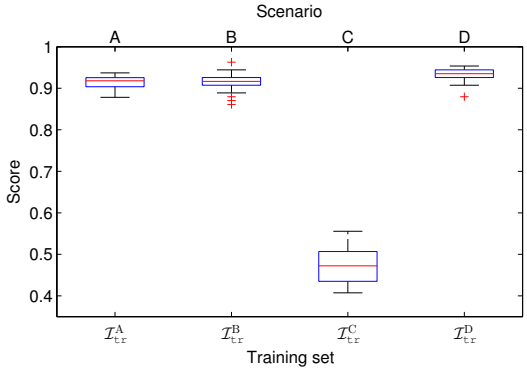
(a) Scenario A: $\mathcal{I}_{\text{tr}}^{\text{A}}$ *(b) Scenario B: $\mathcal{I}_{\text{tr}}^{\text{B}}$*

(c) Scenario C: $\mathcal{I}_{\text{tr}}^{\text{C}}$ *(d) Scenario D: $\mathcal{I}_{\text{tr}}^{\text{D}}$*

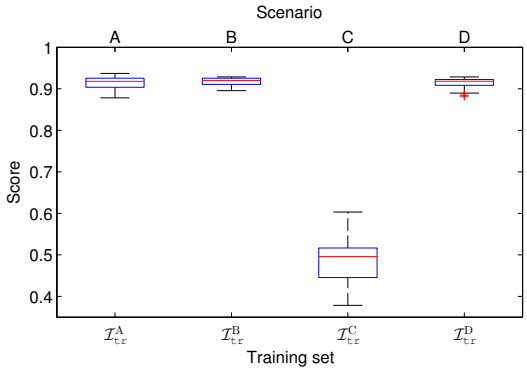
Figure 3.12: Four different scenarios: in scenarios (b)-(c)-(d) features are extracted from square blocks, green blocks come from pristine images, red blocks from fake ones. Black shapes inside pictures/blocks mark tampered regions, while the assigned block label is written underneath it.
In scenario (a), instead, $\mathcal{I}_{\text{tr}}^{\text{A}}$ is composed by full images.

- *Phase 1 training set*: Figure 3.13b shows results obtained training the SVMs in the four scenarios, and testing them on the features obtained from the whole images (not just blocks) of the Phase 1 training set (i.e., $\mathcal{I}_{\text{tr}}$).
- *Phase 1 test set*: Figure 3.13c shows results obtained training the SVMs in the four scenarios and testing them using the features extracted from the whole images (not just blocks) of the Phase 1 test set (i.e., $\mathcal{I}_{\text{ts}}^{1}$).

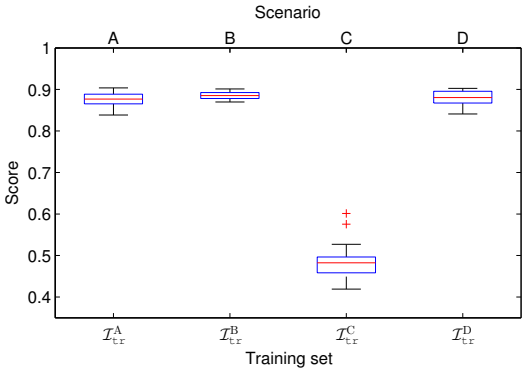
These results clearly show that the algorithm is able to discriminate between the two classes with a high score in all the scenarios, excluding Scenario C. This means that features extracted from images, or blocks, coming from *fake* and *pristine* images respectively, are well separated in the feature space: for this reason the SVMs can be used to separate the two classes. On the other hand, analyzing what happens in Scenario C, we notice that the SVMs fail in separating the two classes even on the training set. This means that, independently from the fact that blocks belong to one class contain a tampered region, the classifier cannot distinguish them from the others.



(a) Scores obtained by testing on $\mathcal{I}^{k}_{tr}$: that is, we train on the training set of each Scenario and we classify it.



(b) Scores obtained by testing on $\mathcal{I}_{tr}$: that is, we train on the training set of each Scenario and we apply learned classifiers on the full images in the Challenge training set.



(c) Scores obtained by testing on $\mathcal{I}^{1}_{ts}$: that is, we train on the training set of each Scenario and we apply learned classifiers on the full images in the Challenge test set.

Figure 3.13: Scores obtained by testing the scenarios on three different test sets. The boxplots represent scores along submodels.
Notice that, regarding the obtained scores, it is indifferent whether we are examining a set of blocks or a set of full images. What is important is that such blocks, or images, actually are taken from different classes (i.e. some images are fake, some are pristine): this is not true in Scenario C.
In fact, its scores are consistent with the scores obtained by a random classifier: in other terms, the classifiers cannot separate the classes in each training set. Note that Scenario C is the sole whose blocks are extracted from images in a single class (i.e. all fakes).

3.4 Conclusions

In this chapter we have shown some results obtained performing a set of tests with an image tampering detector on the dataset distributed for Phase 1 of the IFS-TC Image Forensics Challenge. These tests have highlighted the fact that images labeled as fake and pristine can be reliably distinguished using machine learning techniques. In particular, we developed a variant of the winning algorithm in the IEEE IFS-TC Challenge, based on feature extraction and simple ensemble classifiers to produce a final decision for each image. However, these techniques tend to learn something *not* directly related to the presence of a tampered region: indeed, when training the reported algorithm using unaltered and forged blocks obtained segmenting fake images, the algorithm fails.

The reason behind this behavior may be attributed to how the dataset has been built: we have proven that images labeled as *fake* differ from *pristine* ones for something more than the mere presence of a tampered region.

This could be attributed to the fact that large part of the *fake* images might have been obtained using the same editing software: it is possible that this editing software performs some characteristic operations on the image pixels, probably related to color transformation, independently from the fact that the image has been modified or not.

To verify this conjecture, a deeper analysis would be required, starting instead from a non-blind dataset where *any* detail related to the images is known, spanning from the cameras which took the photographs to the program eventually used for saving images, or the interpolation algorithm which has been used during the resizing phase.

From these tests we can conclude that preparing such a huge image corpus is far from being an easy task, and even choices that in principle might seem insignificant, can make a huge difference on the final result.

Chapter 4

Image tampering localization

In this Chapter, we tackle the Localization problem: that is, given a image which has been modified, the goal is to localize the attacked region in a blind fashion. To this purpose, we developed a novel tampering localization algorithm, and we validate its performance on the Challenge dataset.

Available results in the Localization phase also make use of the official Challenge submission system: only one submission per day is scored, randomly evaluated on a subset of the Challenge Phase 2 test set $\mathcal{I}^2_{\text{ts}}$.

To approach the Localization problem, we combine three techniques, each one revealing diverse information on the images: i) PatchMatch [23]; ii) Near-Duplicate analysis; iii) a novel take on a PRNU-based approach. For each analyzed image, every technique either fails or outputs a binary mask, showing which regions have been classified has been tampered or not.

We begin the chapter by describing each technique separately. Each description is followed by results obtained both on the training set and on the Challenge test set, using the said submission system.

In particular, PatchMatch operates just on a single image, and detects similar regions peculiar to copy-move attacks. The output of PatchMatch is often ambiguous, since it cannot recognize whether any detected region has been used as a source or a destination in the copying process.

Near-Duplicate analysis is a technique which, given a set of images, reveals those who are almost identical to each other. It is a much stronger technique than the other two, since it exploits a redundancy shared by this dataset and by those built by attacks orchestrated using external content which is available to the analyst. Near-Duplicate analysis works by comparing pairs of images, isolating the differences, and combining them across all similar images in the dataset. This approach can suffer from ambiguity problems, too: however, in some cases, we are able to correctly extract the ground

truth mask. Compared to the other two methods, Near-Duplicate analysis is much more powerful, since it is able to precisely segment tampered regions in images; the drawback is that mask availability is scarce, as it necessarily requires to be able to find at least one similar picture for each image.

The third technique which is employed for Localization, has been inspired by the same PRNU-based framework presented in 2.3.1. However, instead of thresholding the PCE surface to decide whether a given region is pristine or not, we form our decision by looking for the best match between the image noise and a PRNU fingerprint. The former is extracted from the examined image, while estimation of the latter involves a great deal of computing.

In fact, as noted in Subsection 2.3.2, to be able to extract PRNU fingerprints we first need to recognize the cameras which have taken the photographs in the Challenge dataset. To this purpose, we describe a simple clustering algorithm which does not rely on any prior information: such clusters should represent images taken by a same camera. With these clusters at hand, under several hypotheses we are able to extract PRNU fingerprints from each cluster.

Next, we finally detail our PRNU-based method to uncover local tampering, and we compare it with the common approach found in literature and carried out in the winning submission.

Since PRNU methods require several hypotheses, we proceed with their discussion and their selective weakening. To this purpose, we also make use of an additional dataset (the *Sensor Attribution Dataset*, hereby SAD) to establish another clustering: since this dataset contains images taken by some of the source cameras in the Challenge, the goal is to obtain a clustering closer to the ground truth. The SAD has been released unofficially to us by the Organizers, and is not meant to be distributed to any of the Challenge competitors. We remark that the SAD is used only for verification purposes. Finally, we compare our simple clustering with the one obtained from the SAD, with the aim to discuss whether we were able to effectively obtain meaningful clusters.

To finish, we design a simple framework to merge the masks obtained for each image by each technique, and to deal the case whether one (or more) of the three methods fails. Fused masks are, then, used to evaluate the performance of each technique and their combinations, in the Challenge setting.

To conclude the chapter, as previously announced in Chapter 3, we also briefly investigate an adaptation of the feature-based detector developed for the Detection problem (see Section 3.1). This approach has also been followed by the winning Challenge submission, cited in [47].

Due to the high score achieved in the Challenge, the approach hereby pre-

sented has been submitted to IEEE International Workshop on Information Forensics and Security (WIFS 2014)[1] as a conference paper [51], and is now under review.

4.1 Proposed algorithm

Our proposed algorithm outputs up to three masks per image, one per technique: they are labeled as M_{PM} (produced by PatchMatch), M_{PRNU} (produced by the novel approach on PRNU) and M_{ND} (produced by near-duplicate analysis). These masks will be fused together, in order to converge to a single decision: the resulting mask is labeled M_{fus}. The set of all M_{fus} will be then used for Challenge scoring.

Notice that M_{PRNU} and M_{PM}, being block based, are actually defined only on center pixels of each block: to bring them to the resolution of the full image, they undergo an interpolation process, discussed in Subsection 4.2.1.

4.1.1 Notation

In the Localization phase, we will mostly work on one image at a time. The image under analysis will be denoted with the symbol I: whenever we need to span across a set $\mathcal{S}$ of images, the generic one will be denoted with I_n, where $n \in \mathcal{S}$.

A mask is associated to each image: if $\mathcal{S} = \mathcal{I}_{\text{tr}}$, the ground truth mask for the n-th image will be denoted as M_n^{GT}, eventually dropping index n whether it is not relevant. We remind that each mask is a set of black and white pixels, where "0" (black) denotes that said pixel is forged, while "1" (white) indicates that the pixel is pristine.

In this section we will often split an image into N blocks whose centers are located at pixels (i_b, j_b). Whenever we obtain a scalar value for each block b, we collect all outputs into a vector. As a slight abuse of notation, to simplify the account we will interchangeably swap the block index b with the corresponding center pixel of coordinate (i_b, j_b), whenever it is unambiguous to do so.

4.1.2 PRNU

In this section we focus on the PRNU-based method we developed to estimate a tampering mask. First we tackle the problem of image clustering for PRNU

[1] `http://ieeewifs.org/`

estimation. Then, given the extracted PRNU, we focus on the localization algorithm.

For the sake of clarity, we assume that all the images share the same size. However, the proposed PRNU-based algorithm can be easily adapted to work on images of different size by simply resizing images beforehand. However, this step would uselessly make the algorithm hard to read, not adding much information on the method. Moreover, considering that the Challenge dataset is strongly biased towards 1024×768 pixel wide images (see Figure 4.8), one might just use our PRNU-based as described for images sharing the same size, still achieving very high accuracy results.

We recall the framework introduced in Section 2.3.1 for general PRNU-based forensic methods.

PRNU clustering

In order to compute a PRNU mask, we need to group each image in the dataset according to the cameras which took it. To do so, we applied the camera attribution procedure to the dataset (see Subsection 2.3.1).

Let $I_n, I_m \in \mathcal{I}$ be two images with the same size, and let W_n, W_m be the respective noise residuals as in Equation (2.7).

For each pair of noise residuals (W_n, W_m) we computed the PCE between them using Equation (2.3.1), collecting the results in a symmetric matrix:

$$\mathbf{P} := \left\{\mathbf{P}(n,m)\right\}_{n,m} = \left\{PCE\left(W_n, W_m\right)\right\}_{n,m} \tag{4.1}$$

Next, we thresholded the PCE matrix $\mathbf{P}$ with a threshold $\tau = 50$, obtaining a binary matrix

$$\overline{\mathbf{P}} := \left\{\overline{\mathbf{P}}(n,m)\right\}_{n,m} = \left\{\mathbb{1}_{\{\mathbf{P}(n,m) > \tau\}}(n,m)\right\}_{n,m} \tag{4.2}$$

This could be enough to initiate a clustering over the dataset. However, reminding what we said in Subsection 4.1.2, we also computed the offset between each pair of noise residuals (i.e. each images) and we selected only images which are aligned (i.e. the reciprocal offset is $(0,0)$). The reason behind this is both to forbid cropping and to make sure that the overlapping area between residuals is as large as possible, thus ensuring a good PRNU estimation. To do so, we maximized the normalized cross-correlation (see Equation (2.12)). In particular, for each pair of noise residuals W_n, W_m we computed the L^2 norm of the offset. Repeating this operation across all pairs, we collected them in a matrix $\mathbf{O}$, whose its generic entry is

$$\mathbf{O}(n,m) := \left\| \arg\max_{i,j} \mathrm{NCC}\left[W_n, W_m\right](i,j) \right\| \tag{4.3}$$

To select aligned images, we marked those pairs whose reciprocal offset is $(0,0)$. This can be expressed through a binary matrix

$$\overline{\mathbf{O}} := \left\{\overline{\mathbf{O}}(n,m)\right\}_{n,m} = \left\{\mathbb{1}_{\{\mathbf{O}(n,m)=0\}}(n,m)\right\}_{n,m} \tag{4.4}$$

Both $\overline{\mathbf{O}}$ and $\overline{\mathbf{P}}$ are then element-wise multiplied in order to produce the matrix on which the clustering is based on, labeled $\overline{\mathbf{D}} = \overline{\mathbf{O}} \times \overline{\mathbf{P}}$. Each non-zero entry, therefore, identifies pairs of images which are associated to the same camera, i.e. they match both in PRNU content and PRNU offset.

We can now proceed to form our clusters: this is done by aggregating each pair of images whose threshold is reached. We can represent the clustering process by an undirected graph $\mathscr{G} = (V,E)$, where the vertex set V is the set of images which have been considered for the clustering, and the edges E are described using $\overline{\mathbf{D}}$ as the graph adjacency matrix. The resulting K clusters are expressed by the connected components of $\mathscr{G}$.

Finally, we proceed with computing the k-th camera PRNU fingerprint $\widehat{K}_k$ from each one of the obtained clusters, and the clustering algorithm stops.

We note that, at this point, the clustering should continue to obtain much better estimates of all fingerprints: however, as this process is extremely time consuming and very hard on computational costs, we decided to trade off accuracy for a very coarse clustering.

PRNU forgery localization

With the cluster fingerprints at hand, we can now match each image with its own alleged PRNU. The method we propose is based on the same rationale as Subsection 2.3.1, but we also exploit information given by blocks correlation offsets.

Let us consider an image I, and let $\widehat{K}$ be the estimated PRNU that we associate to the camera which authored I. Let K be the real PRNU associated to the real camera. In general, simple PRNU-based approaches require two hypotheses:

Hypothesis I (No scaling). *Each image I has not been resized if compared to K.*

Hypothesis II (No cropping). *Each image I has not been cropped if compared to K.*

Hyp. I and Hyp. II together imply that no scaling or cropping operations have *ever* been applied: as a consequence, images and PRNU fingerprints (both K and their estimates $\widehat{K}$) are pixel-wise aligned.

First, suppose for now that Hyp. I and Hyp. II hold: as a consequence, we can match any pixel in I with the same pixel in $\widehat{K}$. The image I is split into overlapping blocks $\{I\}_{b=1}^{B}$, each one centered on the pixel with coordinates (i_b, j_b). If the image is pristine, each block I^b must pass the PRNU correlation test only when centered on PRNU pixel in coordinates (i_b, j_b).

To verify this condition, we compute the correlation as in Equation (2.14) for each pair of matching blocks, with index b. In doing so, we obtain the 2D map $R(i_b, j_b)$. This operation can be greatly accelerated with the Fast Fourier Transform. In the Fourier domain Equation (2.14) becomes (sometimes named as *phase-correlation*):

$$R_b(i_b, j_b) = \mathscr{F}^{-1}\left\{ \frac{\mathscr{F}\{W^b\}\,\mathscr{F}\{\widehat{K}I\}^*}{\left|\mathscr{F}\{W^b\}\,\mathscr{F}\{\widehat{K}I\}^*\right|} \right\} \qquad \forall b \in \{1,\dots,N\}, \tag{4.5}$$

where $\mathscr{F}$ is the Discrete Fourier Transform (zero-padded if needed), $\mathscr{F}^{-1}$ is its inverse and * denotes the complex conjugate. Notice that R_b is a 2D map, showing the correlation between the PRNU and the b-th block shifted of i_b and j_b pixels in the horizontal and vertical dimensions, respectively. Therefore, we can compute the offset estimate between $\widehat{K}$ and I^b as

$$(\hat{i}_b, \hat{j}_b) \coloneqq \underset{(i_b, j_b)}{\arg\max}\, R_b(i_b, j_b)\,. \tag{4.6}$$

If $(\hat{i}_b, \hat{j}_b) = (i_b, j_b)$, the b-th block is compatible with the underlying PRNU, hence the block is considered pristine. On the other hand, if $(\hat{i}_b, \hat{j}_b) \neq (i_b, j_b)$, the b-th block is not aligned with the PRNU, thus it is considered tampered with. This condition is usually not considered in baseline PRNU-based algorithms, which are based on thresholding PCE (or correlation) values. The proposed PRNU tampering mask is then built as

$$M_{\text{PRNU}} \coloneqq \begin{cases} 1, & \text{if } (\hat{i}_b, \hat{j}_b) = (i_b, j_b), \\ 0, & \text{otherwise}, \end{cases} \tag{4.7}$$

where 1 denotes a pristine pixel, and 0 a forged one.

A byproduct of our proposed algorithm is the *displacement map*: by collecting the LHS of Equation (4.6) across all blocks, we obtain

$$\mathbf{D} \coloneqq \left\{ (\hat{i}_b, \hat{j}_b) \right\}_{b=1}^{N} \tag{4.8}$$

Equivalently, by subtracting the block centers coordinates from $\mathbf{D}$, we obtain the *offset map* $\mathbf{O}$. We also separate the coordinate components obtaining $\mathbf{O}_{\mathrm{y}}$ and $\mathbf{O}_{\mathrm{x}}$:

$$\mathbf{O} \coloneqq \left\{ (\hat{i}_b, \hat{j}_b) - (i_b, j_b) \right\}_{b=1}^{N} \tag{4.9}$$

$$\mathbf{O}_{\mathrm{y}} \coloneqq \left\{ \hat{i}_b - i_b \right\}_{b=1}^{N} \tag{4.10}$$

$$\mathbf{O}_{\mathrm{x}} \coloneqq \left\{ \hat{j}_b - j_b \right\}_{b=1}^{N} \tag{4.11}$$

It is worth noting that the derivation of (4.5) from (2.14) strictly requires that the blocks are zero-padded, in order to correctly perform the linear convolution through the circular one. However, the cross-correlation surface between blocks is sharply peaked when a match is established: in other cases, the cross-correlation is close to 0. Since we seek the correct alignment using the PCE, built specifically to detect such peak, this requirement can be heavily relaxed: as a result, we can zero-pad as much as needed in order be able to perform the matrix element-wise products in (4.5).

On hypotheses

Suppose that Hyp. II does not hold: this implies that I and $\widehat{K}$ are not of the same size. The framework presented in 4.1.2 is still applicable: a match can still be obtained by shifting either I or $\widehat{K}$ by the right amount, and zero-padding in order to match their sizes. Moreover, if any of the I_b or W_b is a matrix of zeros (i.e. the b-th block has been cropped away), the corresponding correlation ρ_b is set to 0 since the NCC is not defined for constant-valued blocks.

Now, suppose instead that Hyp. I does not hold: this implies that I and $\widehat{K}$ are not of the same size, too. However, *no* match can be obtained between I and $\widehat{K}$ either through zero-padding or shifting, since the PRNU fingerprint of a set of images $\{I_n\}_n$ is significantly different from the one which can be extracted from their resized versions $\{\mathscr{R}[I_n]\}_n$ (where $\mathscr{R}$ is a resizing operator). Such problem is addressed by Goljan and Fridrich in [16]. The computational requirements are, however, extremely high since their approach requires the images to be resized by means of a set of resizing operators, subsequently demanding another clustering to be performed.

We will, however, experimentally deal with Hyp. I and Hyp. II in Section 4.2.6 using an additional annotated dataset. In particular, we relax both hypotheses and see what happens if the images underwent through a resize to a single resolution, discussing whether the method is still applicable or not.

4.1.3 PatchMatch

PatchMatch enables to detect whether a small patch (e.g. a 7×7 pixels block) can be replaced with another small patch found in the same image, at a very low computational complexity.

More formally, an image I is split into non overlapping 7×7 pixels blocks $\{I_b\}_{b=1}^{B}$, each one centered on pixel coordinates (i_b, j_b). For each block I_b, PatchMatch returns the block that is most similar to it as

$$\hat{I}_b = \underset{I_\beta \in \mathcal{B}}{\arg\min}\, \mathscr{D}\left(I_b, I_\beta\right)$$

where $\mathscr{D}$ is a certain distance metric (mean squared error in our experiments), and $\mathcal{B}$ is a set of possible $\hat{I}_b$ candidates selected by PatchMatch to avoid full-search and patches too close to I_b.

We then store the information about the matching patches into two matrices:

- **D** is a map of the distances between the centers of each matching pair $\langle I_b, \hat{I}_b \rangle$
- **E** is a map of the Mean Squared Error (MSE) introduced if we actually substitute a patch with its matching one.

These maps are built as

$$\begin{aligned} \mathbf{D}(i_b, j_b) &= \left\| (i_b, j_b) - (\hat{i}_b, \hat{j}_b) \right\| \\ \mathbf{E}(i_b, j_b) &= \mathrm{MSE}\left(I_b, \hat{I}_b\right), \end{aligned}$$

where $\|\cdot\|$ is the L^2-norm, (i_b, j_b) and $(\hat{i}_b, \hat{j}_b)$ are vectors collecting the coordinates of I_b and $\hat{I}_b$ central pixels respectively, and $\mathrm{MSE}(\cdot, \cdot)$ computes the MSE between two patches.

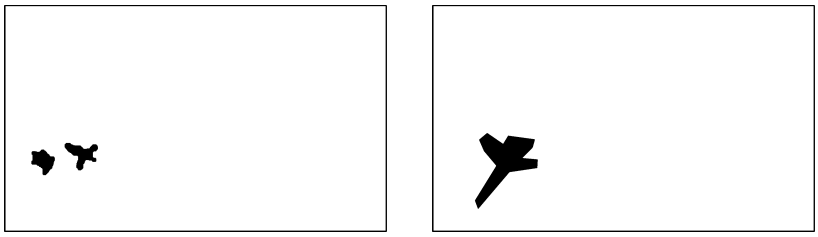
Figure 4.1, Figure 4.2 and Figure 4.3 show three examples of these two maps computed on picture forged using copy-move and Healing Brush. PatchMatch is also adept at recognizing cloning which cannot be distinguished even at a close naked eye examination: an example is reproduced in Figure 4.1.

To compute the binary tampering mask M_{PM}, we segment **D** in regions with the same **D** value. These are areas that can be substituted with pixels at a fixed distance. Among these areas, we select only those larger than a given size (fixing the smallest tampered block we want to detect) and with a low **E** value.

The mask M_{PM} can be optionally refined using morphological operators. It is worth noting that M_{PM} suffers of ambiguity problems when copy-move

attack is used, i.e. it is not possible to disambiguate between the original and copied objects (see Figure 4.2). If more sophisticated attacks are used (e.g. Healing Brush), this problem is less pronounced (see Figure 4.3).

(a) image *(b)* **E** *(c)* **D**

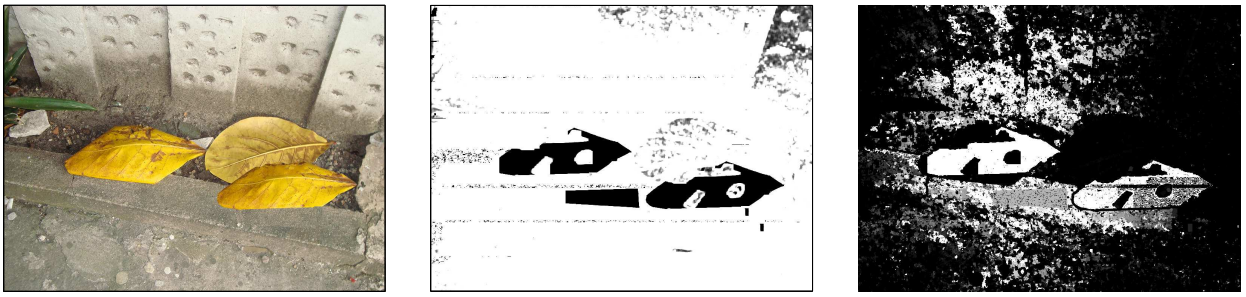


(d) M_{PM} *(e) ground truth*

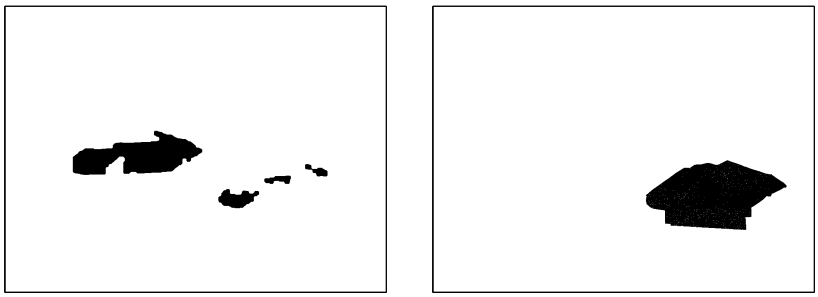
Figure 4.1: **PatchMatch on copy-move**. *A forged image (a),* **E** *(b),* **D** *(c),* M_{PM} *obtained with the proposed approach (d) and the ground truth mask (e). Dark colors represent low values. PatchMatch is adept at recognizing cloning: in particular, this image cannot be classified as being fake even by close inspectioning.*

4.1.4 Near-Duplicate analysis

When dealing with user-generated content distributed online, forged objects are seldom created starting from undistributed original material [52]. In fact, a common image tampering pipeline is to collect and reuse pictures found on media sharing platforms. A typical example is that of image copy-paste forgery operated to substitute the face of a person (e.g., a friend of the forgery creator) with that of another (e.g., a famous artist). It is then possible to search for near-duplicate copies of the image under analysis (i.e. versions of the same image differing due to processing operations, or pictures of the same scene captured from a slightly different point of view) and compare them to find the differences. This search can be either performed via Web crawling, or in the dataset under analysis. An example of a picture coming from the Challenge dataset and a near-duplicate version found online is shown in Figure 4.4.



(a) image (b) **E** (c) **D**



(d) M_{PM} (e) ground truth

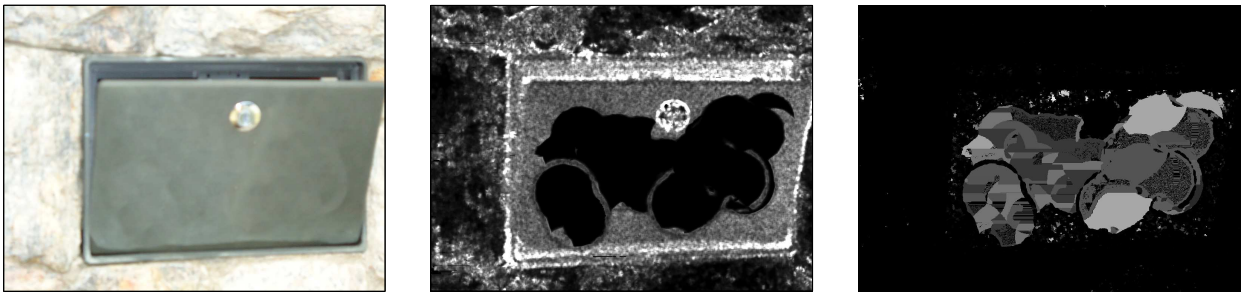
Figure 4.2: **PatchMatch on copy-move**. *A forged image (a),* **E** *(b),* **D** *(c),* M_{PM} *obtained with the proposed approach (d) and the ground truth mask (e). Dark colors represent low values. Here, copy-move has been performed: as a result,* M_{PM} *needs to be disambiguated.*

Robust hashes

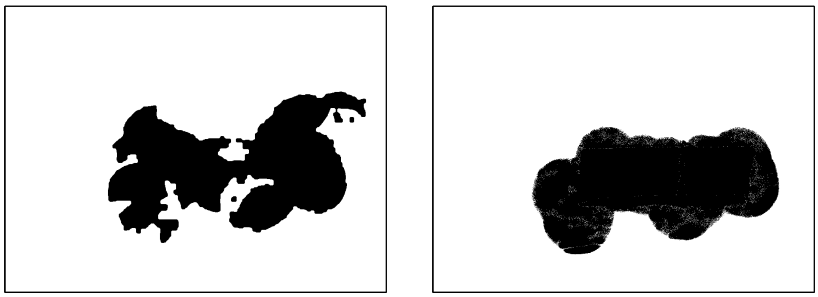
The idea of studying the relationship between pairs of near-duplicate images to find which one has possibly been used to generate the others is at the base of the image phylogeny research field [53]–[55].

Starting from this idea, we propose a near-duplicate-based image tampering localization approach.

The first step consists in determining which images are actually near-duplicates. To this purpose, let us consider a set of images to analyze. We describe each image by means of a *robust hash*, obtained modifying the hash proposed for near-duplicate video matching in [52]. To build the hash, we resize each image I_n to a fixed dimension (256×256 pixels in our experiments). We then compute $\mathbf{Y}_n$ as the 2D Discrete Cosine Transform (DCT) of the resized image. We select a given number of DCT coefficients (in our experiments 256 coefficients $\mathbf{Y}_n(i,j)$, $i \in \{2,\dots,17\}, j \in \{2,\dots,17\}$ discarding horizontal and vertical components whose $i = 1$ or $j = 1$). The selected coefficients are binarized with respect to their median value to obtain the binary hash $\mathbf{h}_n$ (a 256



(a) image (b) **E** (c) **D**



(d) M_{PM} (e) ground truth

Figure 4.3: **PatchMatch on Healing Brush**. *A forged image (a),* **E** *(b),* **D** *(c),* M_{PM} *obtained with the proposed approach (d) and the ground truth mask (e). Dark colors represent low values. Here, Healing Brush has been used, therefore* M_{PM} *mostly agrees with the ground truth.*

bit string in our case, composed by 128 zeros and 128 ones). Hashes are then pairwise compared by computing hamming distance between each pair. If this distance is below a threshold (4 in our experiments), the images related to the compared hashes are considered near-duplicates.

After we identify a near-duplicate I_m of an image I_n under analysis, we compare them pixel-wise to find the differences. To this purpose, we register I_m to I_n, in order to compensate for geometrical transformations such as cropping and resizing. This is done using SIFT matching as suggested in [53]. Then we subtract I_n to the registered version of I_m, obtaining a difference map. This process is depicted in Figure 4.5.

Notice that differences between I_n and I_m are due to the presence of tampering, the presence of noise introduced by processing operations such as JPEG compression, and errors introduced in the registration step. For this reason, to compute the binary tampering localization mask M_{ND}, we need to apply a thresholding operation to the difference map, and optionally use morphological operations. Figure 4.6 and Figure 4.7 shows two examples of

(a) Dataset (b) Web crawled

Figure 4.4: An image coming from the Challenge dataset (a) and a near-duplicate version of it found online (b).

Figure 4.5: The near-duplicate registration process: we search for a geometrical transformation between I_n and I_m using SIFT matching (on the left) and we obtain the registered images (on the right).

near-duplicate images, and the obtained M_{ND} masks.

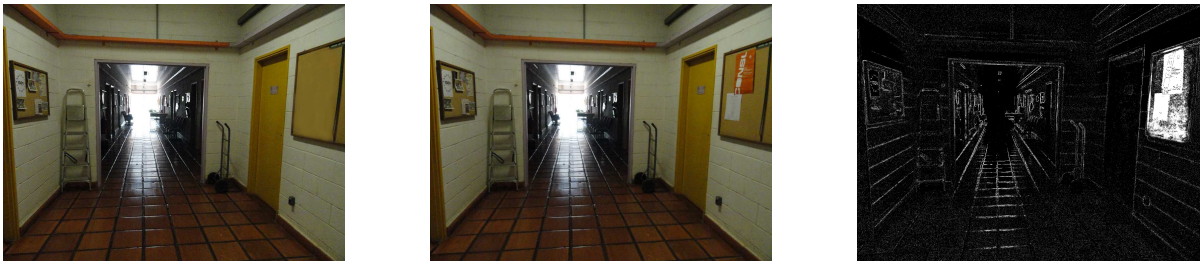
Notice that, if both the compared images contain tampered areas, M_{ND} suffers of ambiguity problems like M_{PM}. If we find K near-duplicates of a reference image, we obtain a set $\{M^k_{\text{ND}}\}_{k=1}^{K}$ of masks (i.e. one for each near-duplicate) that can be merged to solve the ambiguity.

In the Challenge

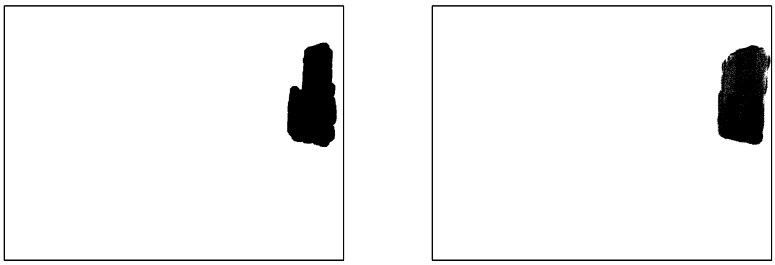
The Challenge dataset presents some redundancy: that is, some images can be traced back as being multiple versions of the same *root* photograph, which can either be or not be available. Hence, by comparing similar images (from now on, *near-duplicates*) inside the Challenge dataset, we can both infer which attack has been performed, and where it has been applied.

To each image $I \in \mathcal{I}$ we can associate a set of K binary masks $\{M^k_{\text{ND}}\}_{k=1}^{K}$, where K is the number of images which are near-duplicates of I: each one of these masks is an estimate of differences between I and its k-th near-duplicate.

From this set, we can synthesize a single binary mask M_{ND}, which is an estimate of differences that are unique to I among all its K near-duplicates. Figure 4.6 shows an example of near-duplicate images, and the obtained M_{ND} mask.



(a) Reference image I *(b) near-duplicate* *(c) difference*



(d) $M_{\text{ND}} = M^1_{\text{ND}}$ *(e) Ground truth M^{GT}*

Figure 4.6: **Near-Duplicates**. *Reference image I_n (a), near-duplicate I_m (b), difference between reference and registered near-duplicate (c), $M_{\text{ND}} = M^1_{\text{ND}}$ (d), and ground truth mask M^{GT}_n. Dark colors represent low values.*

We remark that the method to extract M_{ND} can suffer from ambiguity problems. However, we will exploit other masks M_{PM} and M_{PRNU}, when available, to attempt at disambiguating M_{ND}: this procedure is described in Subsection 4.1.5.

4.1.5 Fusion

To reach the final decision we perform three steps. First we obtain a mask for each image with each technique (i.e. M_{PRNU}, M_{PM} and $\{M^k_{\text{ND}}\}^K_{k=1}$) whenever it is possible to do so. We remind that each one is set to 0 to denote forged pixels, and to 1 to denote pristine pixels according to the respective technique. Afterwards, we try to solve the ambiguity problems which can affect M_{PM} and the near-duplicates. Finally, we merge the obtained disambiguated masks into a single tampering mask M_{fus}.

To this purpose, let us first take into account the inherent properties of

each kind of masks.

Mask properties

M_{PRNU} reveals many kinds of tampering, and revealed forged areas are unambiguous. In other words, we can strongly trust areas detected as having been tampered with. On the other hand, M_{PRNU} hardly reveals forgeries smaller than the block size used to analyze I during the PRNU analysis. For this reason, some forged areas may not be revealed by M_{PRNU}.

M_{PM} is tailored to a specific kind of attack (i.e. copy-move-like), moreover it presents ambiguous regions. The ambiguity is due to the fact that both the original and copied patches are detected (e.g. if an object is replicated, both replicas appear in M_{PM}).

Each M^k_{ND} reveals many kind of forgeries (as M_{PRNU}), nonetheless it suffers from ambiguity problems (as M_{PM}). Indeed, each M^k_{ND} contains information about forgeries on both compared images (i.e. I and its near-duplicate).

Since each mask embeds information that might not be present into the others, a natural method to merge them is the use of the AND operator [47]. However, due to ambiguity in certain masks, this is a suboptimal choice. The pipeline we propose for mask fusion is then the following: we first solve the ambiguity problem whence possible (i.e. for M_{ND}), then we select which masks (i.e. M_{PRNU}, M_{PM} and M_{ND}) to merge with the AND operator, according to a *reliability index* obtained evaluating the masks on the Challenge training set.

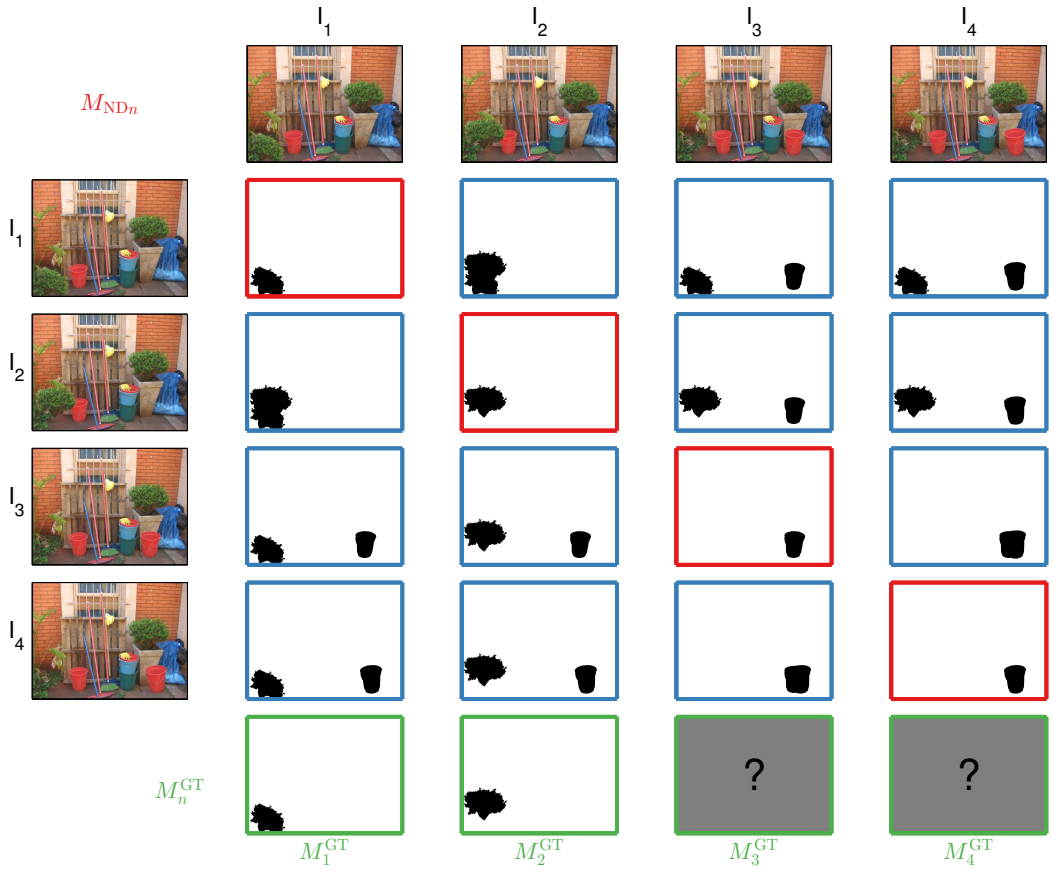
Ambiguity

Suppose that we are seeking an estimate of M^{GT}. When dealing with masks $\{M^k_{\text{ND}}\}_{k=1}^K$, the ambiguity problem can often be solved. Indeed, if $K > 1$, we can resolve this ambiguity by comparing all the $\{M^k_{\text{ND}}\}_{k=1}^K$ masks. We depicted the disambiguation process in Figure 4.7 and Figure 4.17. Each mask reveals the forged area in both I and a near-duplicate version of it, therefore the only forged region that appears in every mask is attributed to I. More formally, we compute the binary mask $M_{\text{ND}} = M^1_{\text{ND}} \vee M^2_{\text{ND}} \vee \ldots \vee M^K_{\text{ND}}$, where $\vee$ is the OR operator.

We note that the Near-Duplicate approach can fail even if it detects a set of near-duplicates. For example, we may obtain a fully white M_{ND} each time a tampering is present in every near-duplicate, so it is impossible to detect by comparing each tuple.

A peculiarity of the Challenge Phase 2 test set $\mathcal{I}^2_{\text{ts}}$ is that all images are fake: to avoid obtaining a fully white M_{ND} (i.e. the image is classified as being pristine), each time it happened we computed $M_{\text{ND}} = M^1_{\text{ND}} \wedge M^2_{\text{ND}} \wedge \ldots \wedge M^K_{\text{ND}}$,

(a) I_1 (b) I_2 (c) I_3 (d) I_4



(e) Images, differences (in blue), ground truths (in green) and disambiguated masks (in red).

Figure 4.7: Near-Duplicate disambiguation with 4 near-duplicates. We notice that the bucket and the plant have been copied across images.
First row and first column: source images, in both $\mathcal{I}_{\mathrm{tr,F}}$ and $\mathcal{I}_{\mathrm{ts}}^2$.
Images with green border: ground truth masks M^{GT} (if present).
Masks with red border: disambiguated $M_{\mathrm{ND}n}$ for n-th image in row.
Masks with blue border: mask differences between row and column images.
The algorithm computes each disambiguated M_{ND} by performing an OR on each row/column.
Notice that the M_{ND} are the same as ground truth masks (in green).

where $\wedge$ is the AND operator. This marks as being fake a region that is larger than the tampering: however, the F_1 scor, employed in the Phase 2 scoring system, favors black regions (fake) rather than white ones.

Another hint for mask disambiguation is to exploit near-duplicates labeled as pristine, for example those in $\mathcal{I}_{\mathrm{tr,P}}$. However, we had only 2 matches between $\mathcal{I}^1_{\mathrm{ts}}$ with known pristine images, therefore we decided not to implement this further optimization.

Two near-duplicate disambiguation

A notable situation where a near-duplicate ambiguity cannot be solved through a single technique, is the case where $K = 2$: that is, only I and one near-duplicate is available. Clearly, we can compare both images, but we cannot distinguish what is pristine from what is not: in fact, M_{ND} simply shows the mutual differences.

One possible way to uncover and remove the ambiguities is to use M_{PM}, if available. Since the shape of forged regions is usually cleaner in M_{ND} rather M_{PRNU} or M_{PM}, an idea to improve M_{PM} is to declare as pristine each fake region in M_{PM} which is also labeled as fake in M_{PM}. However, an initial test with this approach proves to be suboptimal with respect to the Challenge Phase 2 score, as shown in the Appendix. A graphical example of the disambiguation process is reported in Figure 4.18.

Mask selection

We define a set $\mathscr{M}$ of possible masks obtained according to different strategies. In our experiments we considered the following chain:

$$\begin{aligned}\mathscr{M} = \Big\{ & M_{\mathrm{PRNU}},\ M_{\mathrm{PM}},\ M_{\mathrm{ND}},\ M_{\mathrm{PRNU}} \wedge M_{\mathrm{PM}}, \\ & M_{\mathrm{PRNU}} \wedge M_{\mathrm{ND}},\ M_{\mathrm{PM}} \wedge M_{\mathrm{ND}},\ M_{\mathrm{PRNU}} \wedge M_{\mathrm{PM}} \wedge M_{\mathrm{ND}} \Big\} \\ = & \left\{ \mathbf{M}^p \right\}_{p=1}^{7},\end{aligned}$$

where $\wedge$ is the AND operator. Notice that different sets can be defined as well.

We consider a training set $\mathcal{I}_{\mathrm{tr}}$ of forged images $\{I_n\}_{n \in \mathcal{I}_{\mathrm{tr}}}$, whose ground truth tampering mask M_n^{GT} is known. For each image $I_n \in \mathcal{I}_{\mathrm{tr}}$, we compute the set of possible masks $\mathscr{M}_n = \left\{ \mathbf{M}_n^p \right\}_p$. Notice that some strategies cannot be applied on some images (e.g. in case of unknown PRNU), thus resulting in a different cardinality of $\mathscr{M}_n$ for each image.

For each image I_n, we select the strategy that gives the best estimated mask as

$$p_n = \arg\max_p \mathscr{R}\left(\mathbf{M}_n^p, M_n^{\mathrm{GT}} \right),$$

where $\mathscr{R}(\cdot,\cdot)$ is a "metric" of similarity between the compared masks (in our experiments we used the S_2 score adopted for the Image Forensics Challenge described in Section 2.2.1).

We then compute a reliability index C_p for each strategy as

$$C_p := \frac{\left|\{I_n \in \mathcal{I}_{\mathsf{tr}} : p_n = p\}\right|}{\left|\{I_n \in \mathcal{I}_{\mathsf{tr}} : \exists\, \mathbf{M}_n^p\}\right|}, \qquad p \in \{1,\dots,7\},$$

where the numerator represents the number of images for which the p-th strategy is the best one, and the denominator is the number of images on which the p-th strategy could actually be applied.

When a new image $I_m \notin \mathcal{I}_{\mathsf{tr}}$ is evaluated, we use the strategy

$$\overline{p} = \arg\max_p \left(C_p \;\middle|\; \exists\, \mathbf{M}_m^p \right),$$

which is the strategy associated to the mask $\mathbf{M}_m^{\overline{p}}$ (among those that can be computed for I_m) that maximizes the reliability index $C_{\overline{p}}$. We refer to the mask obtained with the fusion procedure as M_{fus}.

Fusion results are reported in Subsection 4.2.5.

4.2 Experimental results

In the localization phase, we will have to deal with images I with their respective masks M. We begin with summarizing their sizes in Figure 4.8: we immediately note that considering only 1024×768 pixel wide images for PRNU processing is not a strong requirement.

4.2.1 Mask interpolation

PatchMatch and PRNU methods produce masks M_{PM} and M_{PRNU} whose coordinates correspond to the center pixels of each block: as a result, their size is very different to the ground truth mask M^{GT}.

To obtain an image mask from M_{PM} (and M_{PRNU}) we first applied a bilinear interpolation operator, where each entry in M_{PM} (and M_{PRNU}) is matched with the center pixel of the block in the full image. As for boundary conditions, we used instead a nearest neighbour interpolation: that is, pixels whose coordinates extend farther from the outermost block center, are set to M_{PM} (or M_{PRNU}) as evaluated in the nearest block center. This enables us to correctly estimate forged objects who extend into the scene from the image border, a rather common case inside the Challenge dataset: however, this is not necessarily the optimal choice.

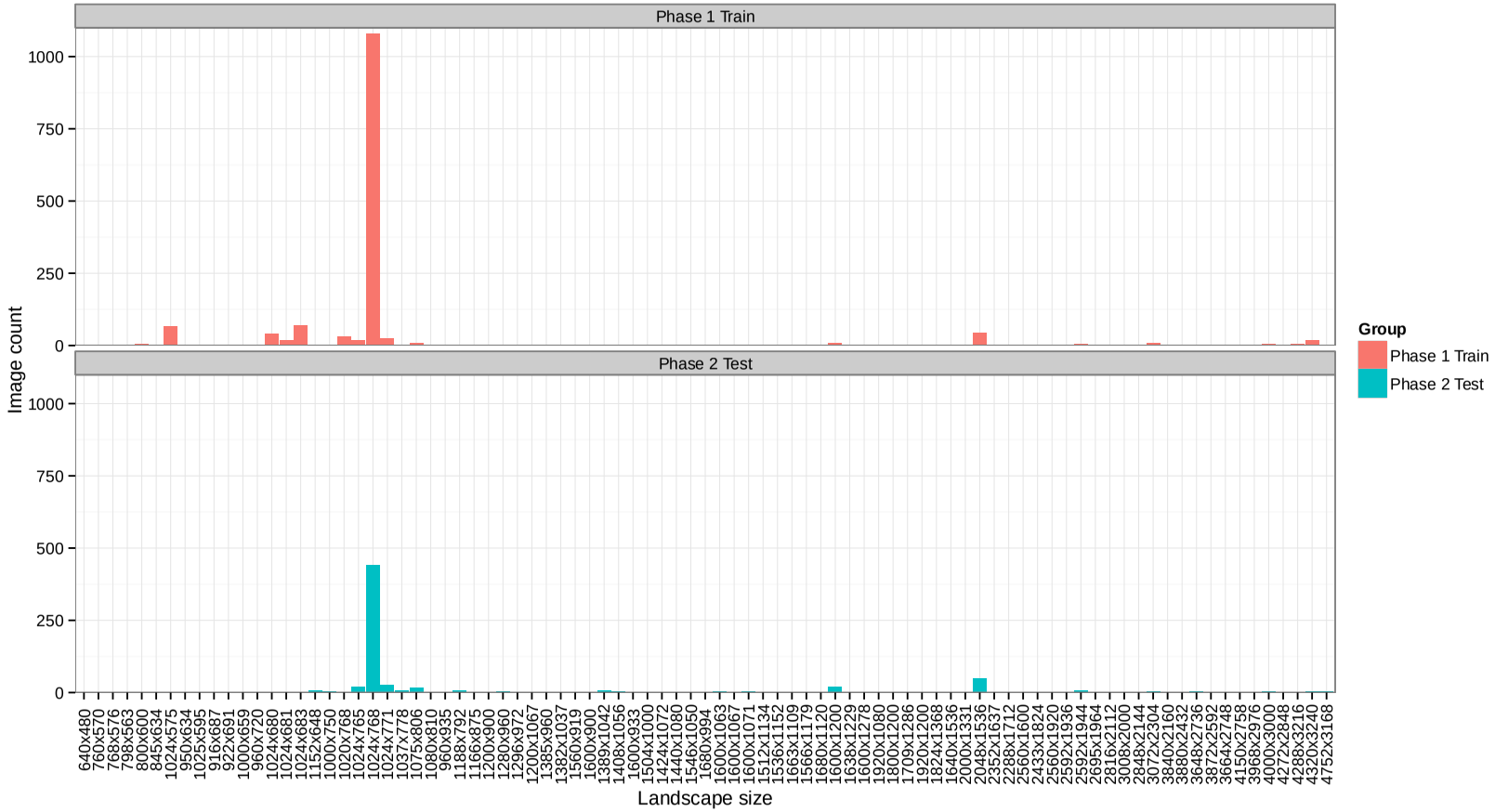


Figure 4.8: Challenge image size distribution in $\mathcal{I}_{\text{tr}}$ and $\mathcal{I}^2_{\text{ts}}$.
To shorten axes, all images have been flipped to landscape orientation, if needed: portrait images constitute a very small part of the displayed datasets.

Finally, as interpolated mask pixels now assume values in $[0,1]$ rather than binary values, we thresholded them using a threshold of 0.5.

4.2.2 Blind PRNU

We applied the PRNU clustering algorithm described in 4.1.2 to the datasets $\mathcal{I}_{\text{tr,F}}$ and $\mathcal{I}^2_{\text{ts}}$, considering only images in Challenge Resolution (see Hyp. III). We report our findings in Figure 4.9 and Figure 4.10.

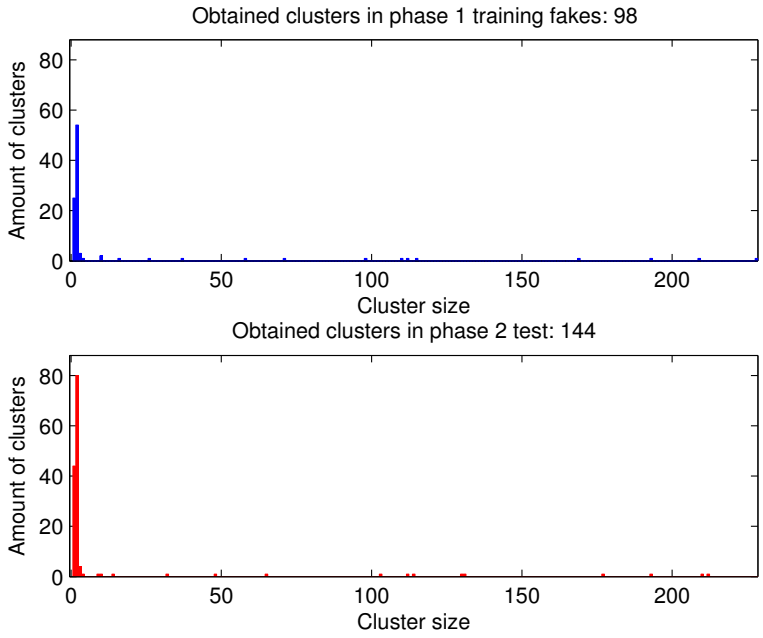


Figure 4.9: Found clusters in $\mathcal{I}_{\text{tr,F}}$ and $\mathcal{I}^2_{\text{ts}}$ (using also $\mathcal{I}_{\text{tr}}$ for clustering). We, respectively,"identified" 98 and 144 cameras, although fingerprints were reliably estimated on about 10 of them: other identified clusters are too small for reliable PRNU extraction.

We were able to attribute a camera, thus indentifying a PRNU fingerprint, for the 45% of images in the former dataset, and more than the 54% in the latter. For these images, we computed two masks, M_{PRNU} and, for comparison, M_{PCE}. M_{PRNU} is computed using the proposed method (as for Equation (4.7)), while M_{PCE} is built using the common approach of thresholding PCE values, followed, e.g. in the winning submission [47].

Recalling the definition of the PCE given in Equation (2.15), from Equation (4.5) we can also compute the PCE between block W_b and the image-PRNU product $I\widehat{K}$, obtaining a PCE surface $P = \{P_b\}_{b=1}^{N}$, where

$$P_b := PCE(\widehat{K}I, W_b) \qquad \forall b \in \{1,\dots,N\} \tag{4.12}$$

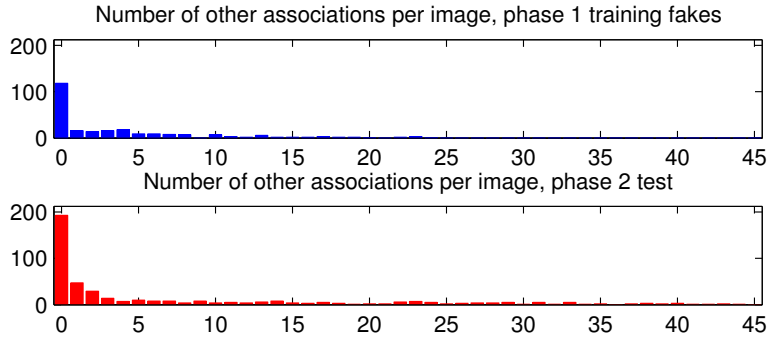


Figure 4.10: Amount of images which can be associated by each image in, respectively, $\mathcal{I}_{\mathrm{tr,F}}$ and $\mathcal{I}^2_{\mathrm{ts}}$. Associable images are, then, clustered together in order to identify a first clustering by camera. Images with 0 associations are, obviously, singletons.

By thresholding P with a threshold τ_b, we obtain the binary mask

$$M_{\mathrm{PCE}} \coloneqq \begin{cases} 1, & \text{if } P_b > \tau_b, \\ 0, & \text{otherwise}, \end{cases} \tag{4.13}$$

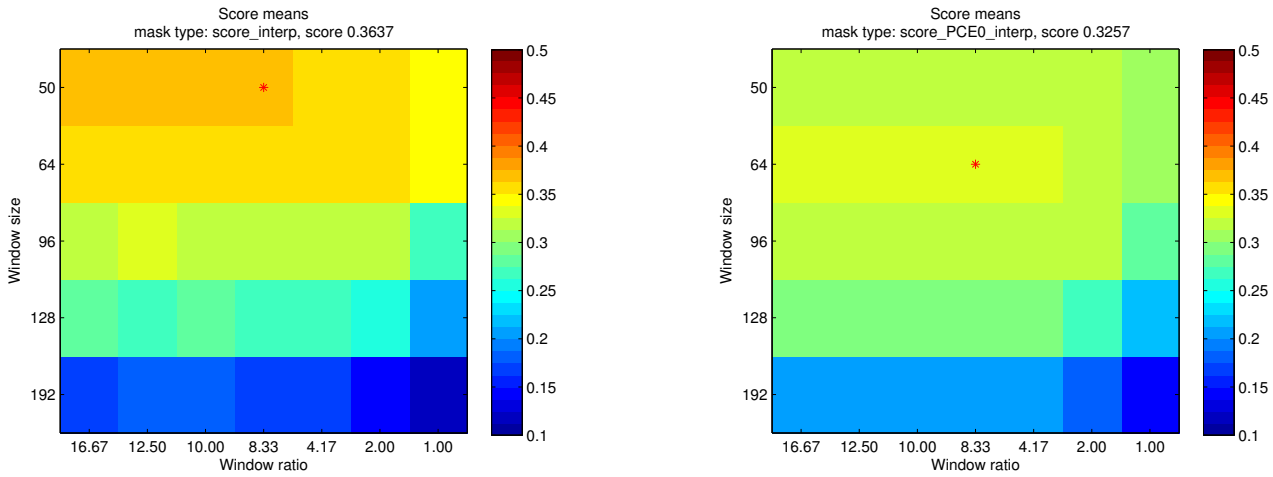
Notice that τ_b is not known a priori, thus calibration is needed: M_{PRNU} does not suffer of this shortcoming.

Block selection

To detect forgeries with PRNU-based methods, the image needs to be first partitioned into blocks. In our trials, we experienced a vast dependence of resulting masks (thus, scores) from the choice of block size and block step (i.e. the horizontal and vertical spacing in pixels between consecutive blocks): as a consequence, we chose the optimal block parameters on the training set $\mathcal{I}_{\mathrm{tr,F}}$ with a 2D grid search. In particular, we chose overlapping square blocks with edge spanning over $[50, 64, 96, 128, 192]$ pixels; instead of selecting a series of window spacings, we set the size-to-step ratio to span over $[16.67, 12.5, 10, 8.33, 4, 17, 2, 1]$.

For example, a block size of 50 with a block step of 50 means that we are partitioning the image into non-overlapping blocks, 50 pixels wide each. To choose a size-to-step ratio of 2 (i.e. block step of 25), instead, produces a grid whose consecutive blocks overlap on half of their area (or a fourth if we choose blocks diagonally adjacent).

For each combination of plot parameters, we computed, whenever possible, M_{PRNU} and M_{PCE} using a suboptimal threshold of $\tau_b = 50$. Finally, we computed the scores over the obtained masks: mean results are shown in Figure 4.11.



(a) Score using M_{PRNU} masks. *(b) Score using M_{PCE} masks.*

Figure 4.11: Mean score using obtained PRNU masks as a function of block parameters. The $*$ marks the best combination, whose corresponding score is reported in the image title.

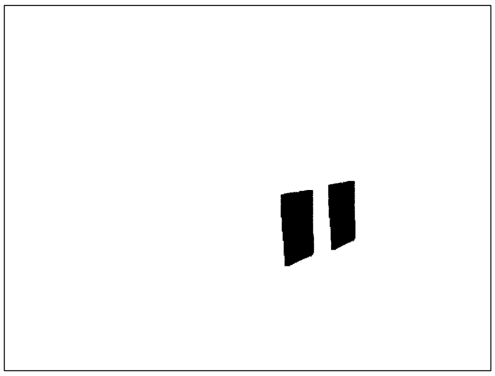
In Figure 4.12 and Figure 4.13 we show some examples of the obtained M_{PRNU} masks, with corresponding M_{PCE} and its companion PCE surface P. In particular, note how in Figure 4.12 the PCE and the correlation surfaces P and R exhibit various content-related structures, rendering difficult the operation of thresholding which produces M_{PCE}. The PRNU mask M_{PRNU}, instead, is clean and crisp, with *no* processing whatsoever.

Notice also that in Figure 4.13, our approach revealed that a copy-move operation has been performed: from subplots (g) and (h), we read that the tampered area correlates with the real PRNU, shifted right by 20 pixels. This means that the road has been translated to the left by 20 pixels. No further processing has been performed on the road: as a consequence, M_{PRNU} is uniform on it. M_{PCE}, instead, detects that the road has been modified everywhere, but no further information is available: as an example, an attacker could have removed road marks on this photograph in order to win a lawsuit related to an accident.

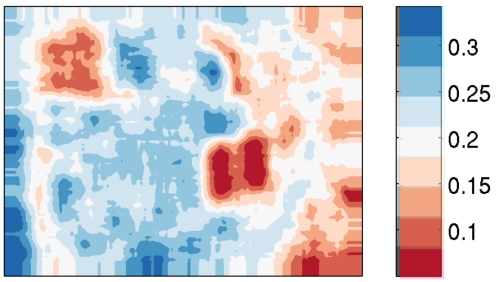
Mask optimization

After having chosen the best block parameters that maximize score on M_{PCE} masks (reported in Figure 4.11b), we optimized M_{PCE} by choosing the best threshold τ_b, using the score as same criterion. As it can be seen in Figure 4.14, the optimal threshold τ_b is much lower than 50 and depends strongly on the

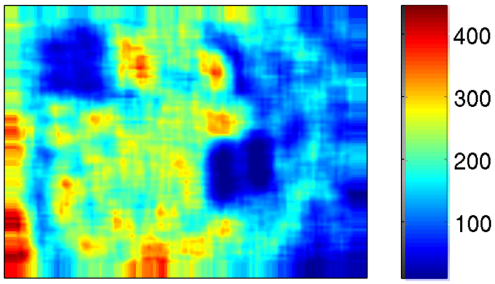
(a) Image I



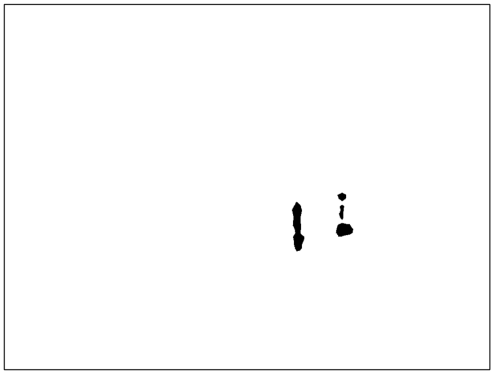
(b) Ground truth M^{GT}



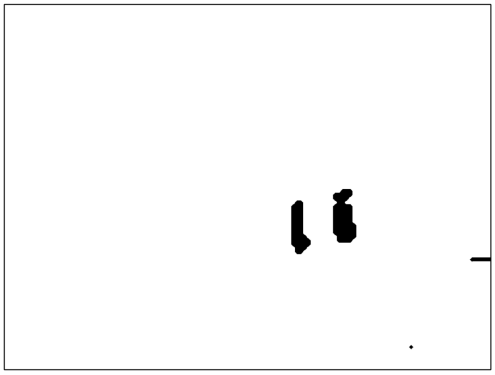
(c) Correlation surface R, interpolated.



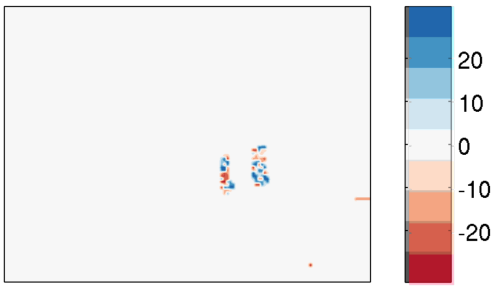
(d) PCE surface P, interpolated.



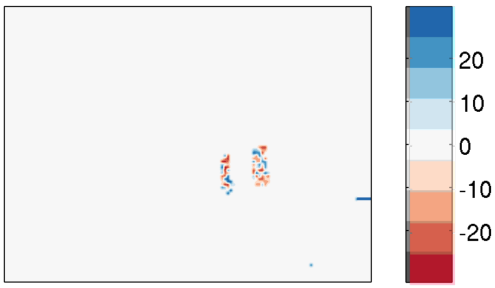
(e) M_{PCE} using $\tau_b = 7$



(f) M_{PRNU}



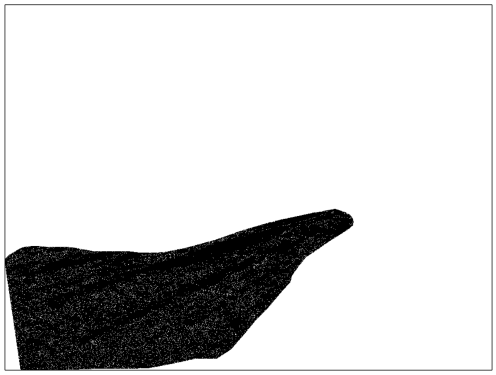
(g) Detected row offset $\mathbf{O}_{\mathrm{y}}$ using M_{PRNU}



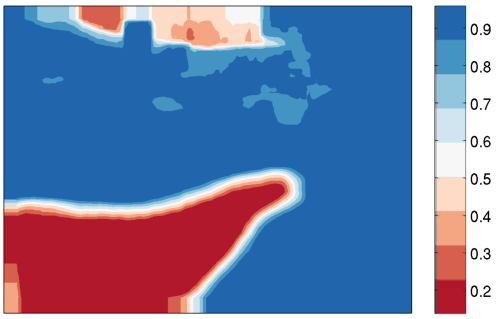
(h) Detected column offset $\mathbf{O}_{\mathrm{x}}$ using M_{PRNU}

Figure 4.12: Sample image with computed masks. Notice how messy is the PCE surface P: given the low PCE values, the top left poster is also a candidate for being a possible fake region. In M_{PRNU}, instead, we obtain a sharp mask with no ambiguities.

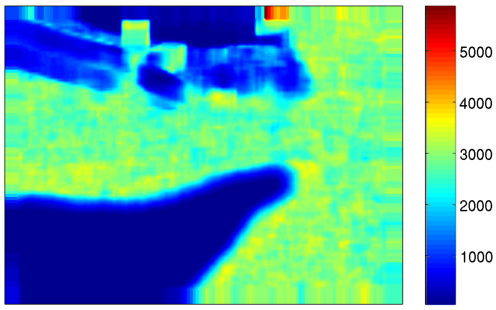
(a) Image I



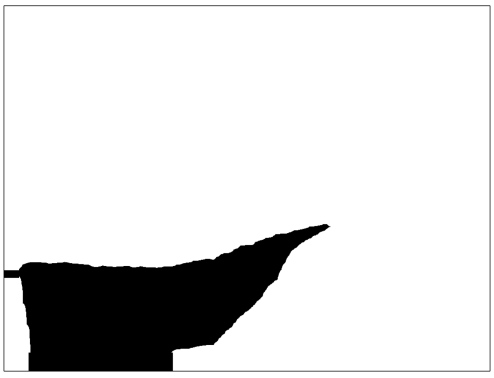
(b) Ground truth M^{GT}



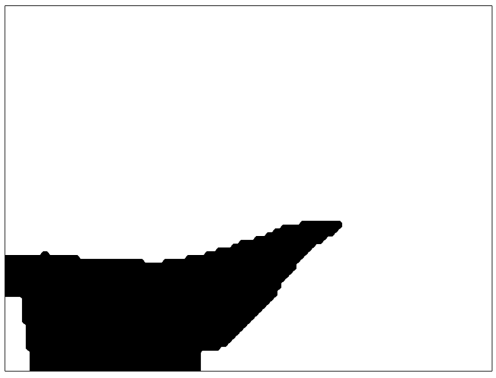
(c) Correlation surface R, interpolated.



(d) PCE surface P, interpolated.



(e) M_{PCE} using $\tau_b = 7$



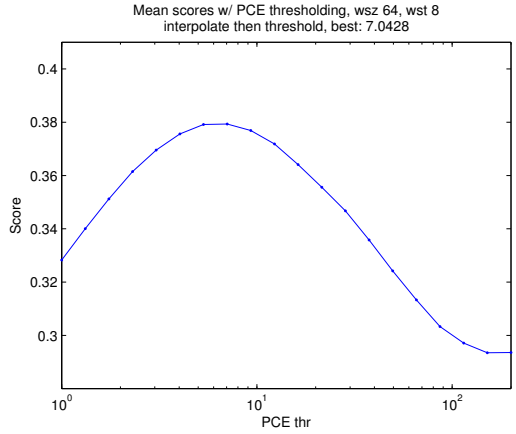
(f) M_{PRNU}

(g) Detected row offset $\mathbf{O}_{\mathrm{y}}$ using M_{PRNU}

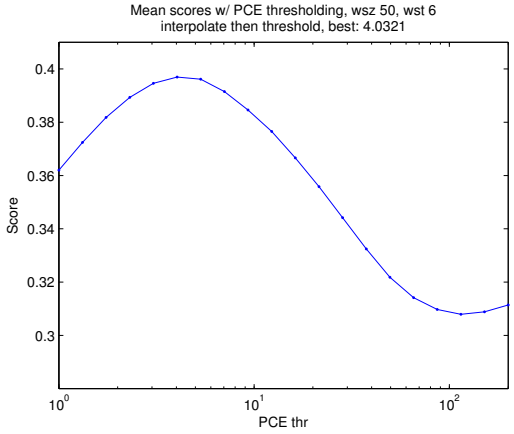
(h) Detected column offset $\mathbf{O}_{\mathrm{x}}$ using M_{PRNU}

Figure 4.13: Sample image with computed masks. Notice that $\mathbf{O}_{\mathrm{y}}$ is fully white (i.e. no row offset is detected), while $\mathbf{O}_{\mathrm{x}}$ is not. This means that the detected region has been captured by the identified PRNU, but has been horizontally translated. We can also read the translation offset from $\mathbf{O}_{\mathrm{x}}$, i.e. 20 pixels.

block parameters: however, due to the lack of time, we did not analyze this relation.



(a) Score using M_{PCE} masks. Chosen block parameters maximize the score at coarse level.



(b) Score using M_{PCE} masks. Chosen block parameters do not maximize the score at coarse level, but they do once level τ_b is optimized.

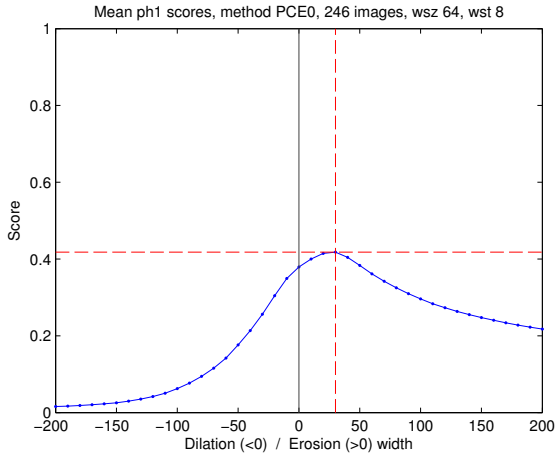
Figure 4.14: Dependence of score using M_{PCE} as a function of PCE threshold τ_b.

Another kind of optimization that can be performed on masks is morphological processing: it consists in enlarging/reducing black parts of M_{PCE} and M_{PRNU} to enhance the score. As Figure 4.15 shows, in general, the score increases by eroding each mask with a disk of 20 pixels of diameter. However, this step is computationally heavy, and we conjecture that its dependence on image content is strong: a much deeper analysis would be required. Also observe that a score increase with erosion is expected: F_1 score favors black regions over white ones, and we often localize altered regions which are smaller than the corresponding zones in the ground truth masks.

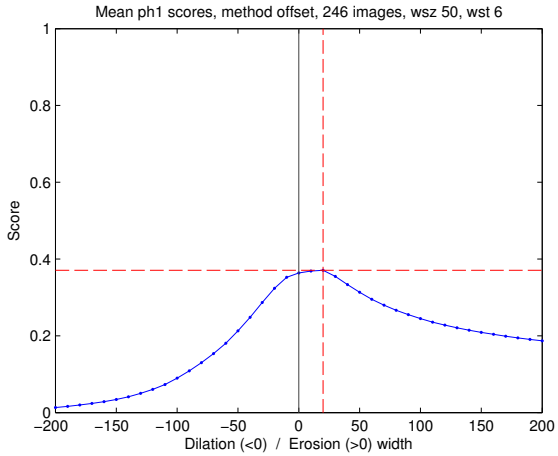
A word of caution: obtained scores on each mask vary wildly, therefore minute differences in scores, such as those reported in this subsection, may be not statistically significant. The large variance can be appreciated in Figure 4.16, which reports the full score distribution for each choice of parameter.

Results on training set

In Table 4.1 we report some statistics obtained on Phase 1 training dataset $\mathcal{I}_{\mathrm{tr,F}}$. We compared M_{PRNU}, M_{PCE} (with optimized threshold) and M_{PRNU} (with morphological processing): each mask has been obtained using the best block parameters for the chosen method. Recalling 2.2.1, we reported the statistics defined in Table 2.2.

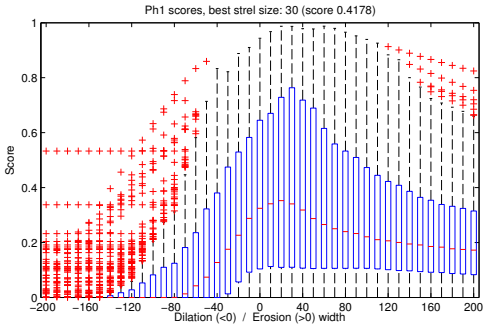


(a) Mean score on dilated/eroded M_{PCE} masks.

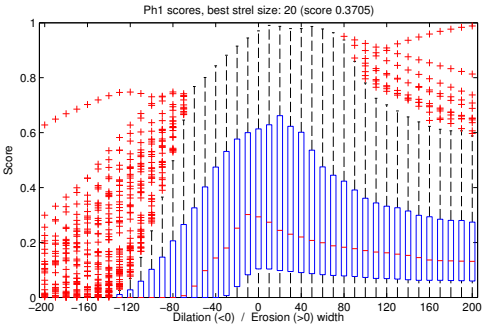


(b) Mean score on dilated/eroded M_{PRNU} masks.

Figure 4.15: Morphological processing. Mean score using obtained PRNU masks. For each plot, we selected the best block parameters according to the respective plot in Figure 4.11. Red marks show the diameter of the best structuring element and the obtained score.



(a) Mean score on dilated/eroded M_{PCE} masks.



(b) Mean score on dilated/eroded M_{PRNU} masks.

Figure 4.16: Morphological processing. Same plot as Figure 4.15: however, we show the full score distribution along all *available masks.*

	TPR (%)	TNR (%)	FPR (%)	FNR (%)	ACC (%)
M_{PCE} (optimized)	61.36	82.51	17.49	38.64	71.93
M_{PRNU}	68.44	74.39	25.61	31.56	71.41
M_{PRNU} (optimized)	82.82	68.45	31.55	17.18	75.63

Table 4.1: Comparison between PRNU-based forgery localization methods.

4.2.3 Near-Duplicates

We extracted all near-duplicates across $\mathcal{I}_{\text{tr,P}}$, $\mathcal{I}_{\text{tr,F}}$ and $\mathcal{I}^2_{\text{ts}}$: as one can see from Table 4.2, near-duplicates constitute a significant part of the Challenge datasets. Most of them, however, can be only paired together, therefore most obtained masks are ambiguous. Nonetheless, this method still achieves a score much higher than the others, normalized on the number of effectively computed masks Table 4.6.

k	0	1	2	3	4
$\mathcal{I}_{\text{tr,F}}$	350	93	5	1	1
$\mathcal{I}^2_{\text{ts}}$	493	186	11	1	9

Table 4.2: Amount of images with k other near-duplicates in $\mathcal{I}_{\text{tr}} \cup \mathcal{I}^2_{\text{ts}}$. An image with 0 near-duplicates is obviously a singleton. E.g. in $\mathcal{I}_{\text{tr,F}}$ there are 93 images sharing a near-duplicate among $\mathcal{I}_{\text{tr}} \cup \mathcal{I}^2_{\text{ts}}$: this also means that some near-duplicates are counted twice.

In Figure 4.7 and Figure 4.17 we show some examples of the algorithm at work. Notice the crispness of the mask borders, thus rendering futile to apply further morphological processing.

4.2.4 PatchMatch

In Figure 4.18 we show an example of a tentative disambiguation of a near-duplicate pair using a PatchMatch mask.

4.2.5 Mask fusion

Using the Challenge training set, we computed the reliability index for every fusion strategy $\{\mathbf{M}^p\}_{p=1}^7$. It is clear from Table 4.3 that the strategies involving only one kind of mask (i.e. M_{ND}, M_{PRNU}, M_{PM}) perform better than mixed strategies.

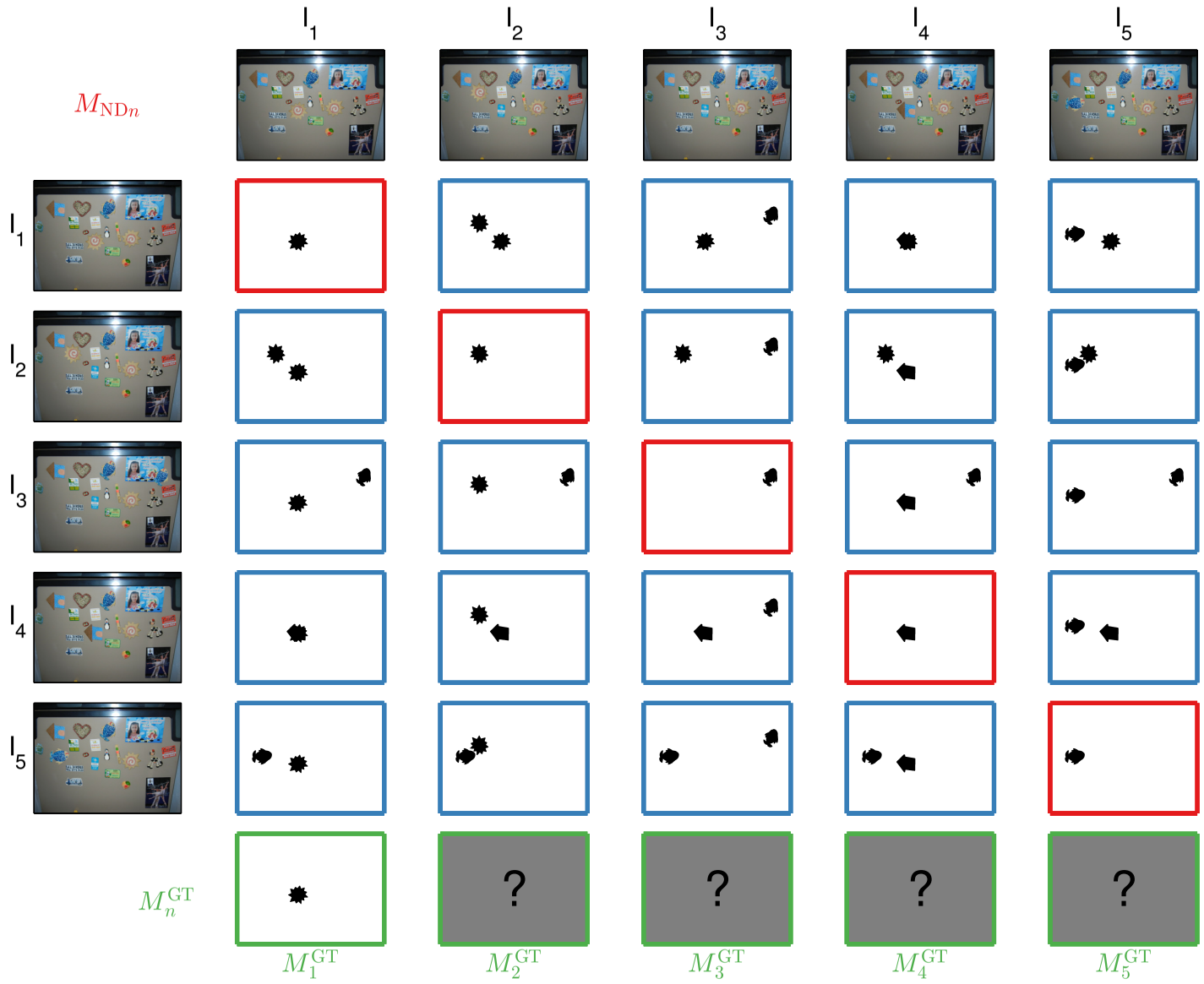
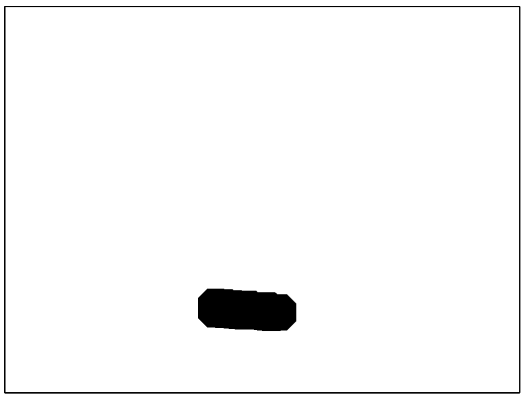


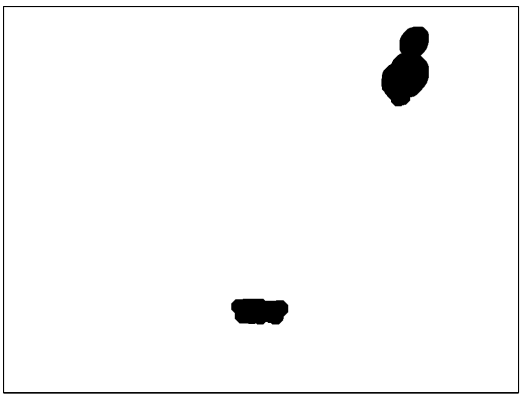
Figure 4.17: Near-Duplicate disambiguation with 5 near-duplicates. We notice that the fridge magnets move across images.
First row and first column: source images, in both $\mathcal{I}_{\mathsf{tr,F}}$ and $\mathcal{I}_{\mathsf{ts}}^2$.
Images with green border: ground truth masks M^{GT} (if present).
Masks with red border: disambiguated $M_{\mathrm{ND}n}$ for n-th image in row.
Masks with blue border: mask differences between row and column images.
The algorithm computes each M_{ND} by performing an OR on each row/column.

(a) Reference image I_1

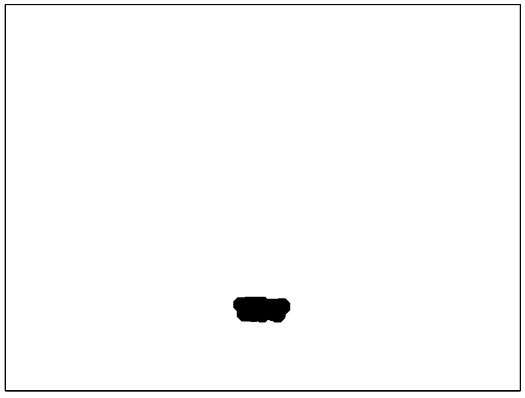
(b) Comparison image I_2



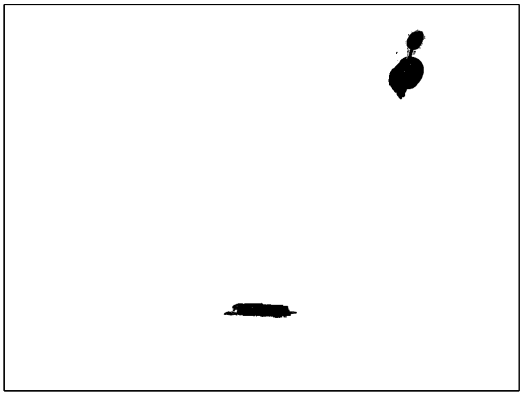
(c) M_{PM} for I_1



(d) Ambiguous M_{ND} for I_1



(e) Disambiguated $M_{\mathrm{ND}1}$ for I_1



(f) M_1^{GT}

Figure 4.18: **ND-PM disambiguation**. *(a) and (b) are two near-duplicate images, labeled I_1 and I_2. (d) and (c) are, respectively, the difference mask between them (i.e. M_{ND}^1) and the PatchMatch mask on I_1.*
By combining M_{PM} and M_{ND}^1 as in 4.1.5, on I_1 we obtain (e) which is different than M_1^{GT}, reported in (f). (In fact, actually, I_2 is pristine.)

$\mathbf{M}^p$	M_{PRNU}	M_{PM}	M_{ND}	$M_{\text{PRNU}} \wedge M_{\text{PM}}$	$M_{\text{PRNU}} \wedge M_{\text{ND}}$	$M_{\text{PM}} \wedge M_{\text{ND}}$	$M_{\text{PRNU}} \wedge M_{\text{PM}} \wedge M_{\text{ND}}$
$C_p(\%)$	23.90	25.19	33.16	1.60	13.74	2.09	0

Table 4.3: Reliability indexes on $\mathcal{I}_{\text{tr,F}}$. Each C_p is computed only on available masks for strategy $\mathbf{M}^p$. One can appreciate how single strategies are favorable over joint ones.

As a consequence, we adopted the following rationale for Phase 2 submissions: for each image $I_n \in \mathcal{I}^2_{\text{ts}}$ we first select the mask which corresponds to best performing strategy among $\mathscr{M}_n$, i.e. M_{ND}. If M_{ND} is not available, we chose, in order of availability, M_{PM} and M_{PRNU}. If neither of these strategies have been selected for image I_n, we select a fully black mask, since a white one would have obtained a score equal to 0, by definition.

In Table 4.4 we report the same statistics as in Table 4.1 using all available techniques. The cardinality of obtained sets is reported, instead, in Table 4.5.

We also submitted the extracted masks using the Challenge submission systems. Selected obtained scores are presented in Table 4.6, while a full log is available in Appendix B.2.

Along the official S_2 score, we also reported a normalized version of S_2, named $\overline{S}_2$, which has been computed by weighted averaging masks effectively submitted: this is a measure of effectiveness of each technique, defined to be independent on the number of available masks. As a consequence, e.g. corrected score for M_{ND} is much higher than other methods: this is due to the fact that M_{ND} availability is scarce.

Score normalization

Suppose that we made a submission to the Challenge official system: i.e. we submitted our guesses on $\mathcal{I}^2_{\text{ts}}$. Let N_0 be the cardinality of $\mathcal{I}^2_{\text{ts}}$, and let S_2 the score returned by the official system for our submission.

Suppose that we submitted a set of N masks in $\mathcal{I}^2_{\text{ts}}$ for a given technique. As only "full" submissions are scored, we need to *pad* the set of submitted masks with other guesses, for example, with N_w white and N_b black masks. We observed that white or black masks are set to have F_1 score of, respectively, 0 (by definition, since no pristine images are present in $\mathcal{I}^2_{\text{ts}}$) and 0.12 (from `submission_001` and experiments on $\mathcal{I}_{\text{tr}}$, as reported in Figure B.1).

As a consequence, neglecting the score randomization, from Equation (2.5) we may write, on average

$$S_2 = \frac{1}{N_0}\left(N\overline{S}_2 + 0 \cdot N_w + 0.12 \cdot N_b\right),$$

hence we define the normalized score $\overline{S}_2$ as

$$\overline{S}_2 := \frac{1}{N}\left(S_2 N_0 - 0.12 N_b\right). \tag{4.14}$$

Technique	Available masks	TPR (%)	TNR (%)	FPR (%)	FNR (%)	ACC (%)	F_1
M_{ND}	100	83.75	99.05	0.95	16.25	97.81	0.7603
M_{PRNU} (opt.)	203	82.82	68.45	31.55	17.18	69.74	0.4514
M_{PRNU}	212	68.44	74.39	25.61	31.56	73.96	0.4165
M_{PM}	204	55.61	94.66	5.34	44.39	91.44	0.4541
M_{fus}	288	67.46	92.13	7.87	32.54	89.67	0.5552

Table 4.4: Comparison between forgery localization methods. Each column is averaged on all available masks for each technique in $\mathcal{I}_{\text{tr,F}}$. Notice how powerful is M_{ND} against all other techniques: near-duplicate information is, in fact, a strong peculiarity of the Challenge dataset. Reported statistics were defined in Table 2.2.

	Available masks	
Mask type	% of $\mathcal{I}_{\text{tr,F}}$	% of $\mathcal{I}^2_{\text{ts}}$
M_{PRNU}	45	54
M_{PM}	45	48
M_{ND}	22	29
M_{fus}	**64**	**66**

Table 4.5: Percentage of computed tampering masks using different methods and datasets.

4.2.6 Non-blind PRNU

As mentioned at the start of Chapter 4, we were also provided upon our request with the *Sensor Attribution Dataset*, hereby shortened as "SAD". The SAD is a list of 35 sets of 20 pristine images each. Every set has been captured by a single camera which might have been used in the Challenge; some sets are extraneous, though.

We exploited the SAD in order both to precisely estimate PRNU fingerprints beforehand, and to assess whether PRNU-based methods were applicable or not.

	Score on $\mathcal{I}_{\text{tr,F}}$		Score on $\mathcal{I}^2_{\text{ts}}$	
	per image	global	per image	global
M_{PRNU}	0.4514	0.2483	0.3317	0.2535
M_{PM}	0.4541	0.2563	0.4190	0.2784
M_{ND}	0.7603	0.2679	0.8448	0.3331
M_{fus}	0.5552	0.3853	0.5669	0.4533

*Table 4.6: S_2 score for the obtained masks. Scores are presented for the whole dataset (*global*, that is, $\mathcal{I}_{\text{tr,F}}$ or $\mathcal{I}^2_{\text{ts}}$) and averaging it only on the number of available masks for the selected technique ($\overline{S}_2$,* per image*). Note that the last one is the highest score which we ever achieved in the Challenge.*

Four PRNU patterns

Instead of simply estimating one fingerprint per camera, we wish to control all cases where a source image I^0 has been resized to the Challenge resolution (1024×768 for the vast majority of the dataset): as a big plus, this would also be a way to deal with Hyp. I, albeit in a less generic context.

For the sake of simplicity, we will refer to 1024×768 pixel wide images as the *Challenge Resolution* (CR).

Let K be the *true* PRNU fingerprint associated to a source image I^0. We relaxed Hyp. I to the following:

Hypothesis III (Scaling and cropping to Challenge Resolution). *Each image $I^0 \in \mathcal{I}_{\text{tr}} \cup \mathcal{I}^2_{\text{ts}}$ has been either resized to CR, or cropped, but not both.*

Let $\mathscr{R}$ be the operator which resizes any source image I^0 to the CR: that is, $I^* = \{I^*(i,j)\}_{i,j} := \mathscr{R}[I^0]$ has the CR $\forall I^0$.
Let $\{I_i\}_{i=1}^{20} \subset \text{SAD}$ be a set of images taken by the k-th camera.

Under Hyp. III, from each $\{I_i\}_{i=1}^{20}$ we extracted four *kinds* of PRNU fingerprints: (see Figure 4.19)

- **Full PRNU**: $\widehat{K}^{\text{F}}$
  $\widehat{K}^{\text{F}}$ is estimated from $\{I_i\}_{i=1}^{20}$ using the procedure described in 2.3.1.

- **Cropped PRNU**: $\widehat{K}^{\text{C}}$
  $\widehat{K}^{\text{C}}$ is estimated from 1024×1024 crops taken from the center from all images in $\{I_i\}_{i=1}^{20}$, using the procedure described in 2.3.1. We assume

that every image in $\mathcal{I}$ contains part of the cropped region.
With this test we simulate the cases where Hyp. I holds but Hyp. II does not.

- **Full resized PRNU**: $\hat{K}^{\mathrm{FR}}$
  $\hat{K}^{\mathrm{FR}}$ is obtained by resizing $\hat{K}^{\mathrm{F}}$ to the CR:

$$\hat{K}^{\mathrm{FR}} := \mathscr{R}[\hat{K}^{\mathrm{F}}]$$

- **Resized PRNU**: $\hat{K}^{\mathrm{R}}$
  We applied the resizing operator $\mathscr{R}$ to each image in $\{I_i\}_{i=1}^{20}$, thus obtaining $\{\mathscr{R}[I_i]\}_{i=1}^{20}$. Next, $\hat{K}^{\mathrm{R}}$ is estimated from $\{\mathscr{R}[I_i]\}_{i=1}^{20}$ using the procedure described in 2.3.1.

For each image I_n, we collect these PRNUs under the symbol $\mathscr{K}_n$:

$$\mathscr{K}_n := \left\{\hat{K}^{\mathrm{F}}, \hat{K}^{\mathrm{C}}, \hat{K}^{\mathrm{FR}}, \hat{K}^{\mathrm{R}}\right\}$$

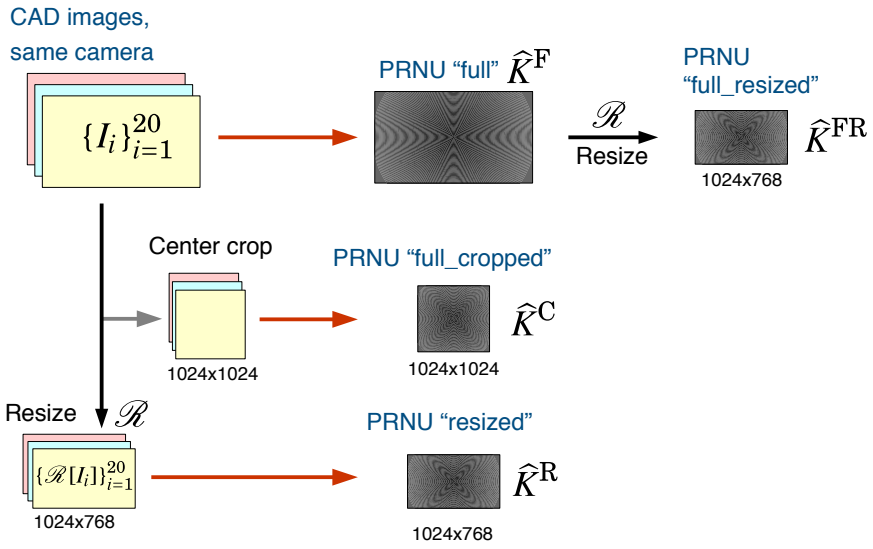


Figure 4.19: The PRNU extraction process.

This procedure has been repeated for each of the 35 cameras in the SAD, thus obtaining $4 \times 35 = 140$ alleged PRNU fingerprints, four per camera, of different nature one another.

We observe that in a fully blind setting, such as the Challenge one, only $\hat{K}^{\mathrm{R}}$ can be estimated, whence the generic PRNU-based localization framework strictly requires $\hat{K}^{\mathrm{F}}$ (see Hyp. I). As a consequence, in the 4.2.6 we first discuss if a PRNU-based forensic approach is applicable by using $\hat{K}^{\mathrm{F}}$ and $\hat{K}^{\mathrm{C}}$,

comparing also the attributed cameras with $\widehat{K}^{\mathrm{FR}}$. Next, in 4.2.6 we will check whether the resizing operator $\mathscr{R}$ commutes with the PRNU extraction: i.e. we compare camera attributions obtained using $\widehat{K}^{\mathrm{FR}}$ and $\widehat{K}^{\mathrm{R}}$. If this assumption is true, we conclude that we can safely apply PRNU-based methods on the entire Challenge set $\mathcal{I}$ if Hyp. III holds.

Non-blind camera attribution

For each one of the 35 cameras, we estimated the aforementioned 4 kinds of PRNU fingerprints, thus obtaining 140 PRNU fingerprints, four per camera. Next, we matched the entired Challenge train set $\mathcal{I}_{\mathsf{tr}}$ and the Phase 2 test set $\mathcal{I}^2_{\mathsf{ts}}$ using the Camera Attribution procedure as in 2.3.1, using a PCE threshold of $\tau = 50$. We represented a sample Camera Attribution procedure on a selected image in Figure 4.20.

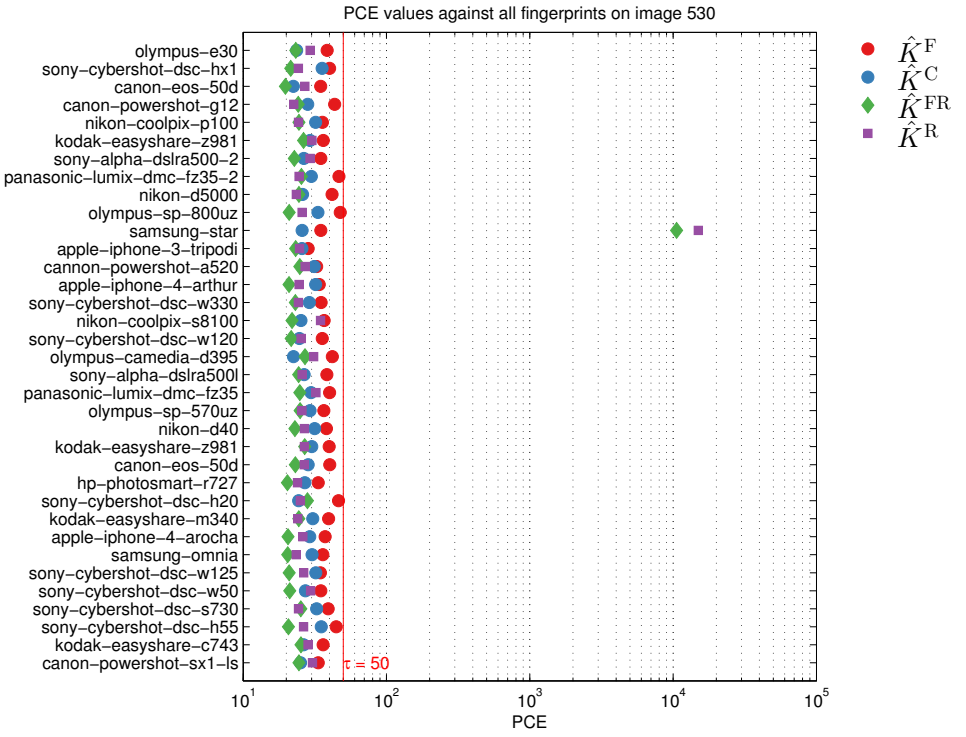


Figure 4.20: Camera Attribution using SAD on image $I_{530} \in \mathcal{I}_{\mathsf{tr}}$: we matched its noise with all available fingerprints extracted all available cameras. The red line is the PCE threshold $\tau = 50$ above which we declare a camera match.
Notice the double match with a PRNU pattern which has been extracted both from resized images and has been resized from full-sized fingerprint: since it is attributed to a single camera, this is a desired situation, as explained in 4.2.6.

We summarize our results in Figure 4.21 and Figure 4.22. The former represents how many images are attributed to each camera, spliced along the four kinds of PRNU. We observe that some cameras have not been matched, and other constitute a significant part of the Challenge dataset, thus rendering possible a good PRNU estimation. From the latter plot, we conclude that we can reliably attribute a camera to at least half of each dataset: by strictly considering a PCE threshold $\tau = 50$, in $\mathcal{I}_{\text{tr}}$ we declare as identified 1126 images over 1500, while in $\mathcal{I}^1_{\text{ts}}$ we identify 528 images over 700.

Blind detectability

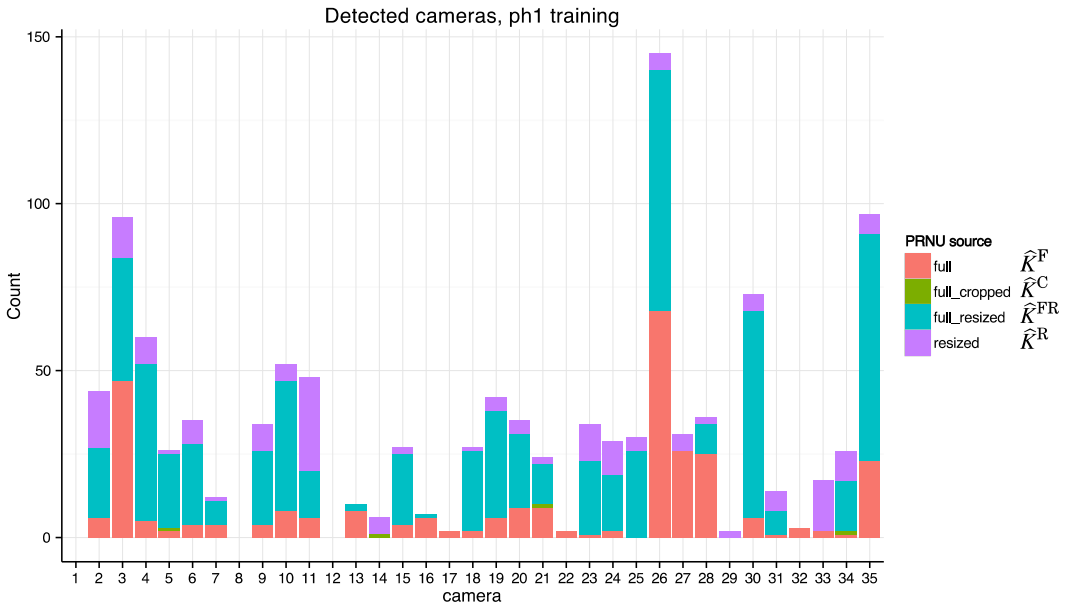
We also analyzed whether a fully blind clustering on these datasets is feasible, on the reasonable assumption that the clustering metric is PCE-based. As such, to extract accurate PRNU fingerprints, a significant amount of images is needed: moreover, since we are in a blind setup, we need to make sure that those images actually have been captured by the same camera. Another reason for this analysis is that PCE decreases whenever an image is re-compressed or gets tampered with, thus decreasing the reliability of each pixel to convey information on PRNU and the source camera.

The detectability plots are reported in Figure 4.23 and Figure 4.24. Each plot is devised to assess whether a blind clustering is possible, and the expected reliability on the obtained PRNU. In these plots we work with groups of images which have been attributed to the same camera; each group is represented on the x-axis by the distribution of the PCE of all images against its PRNU fingerprint, while on the y-axis we report the size of the group. To avoid clutter, instead of representing the full distribution of the PCE, we show with a horizontal line only the first and third quartile along with the group median.

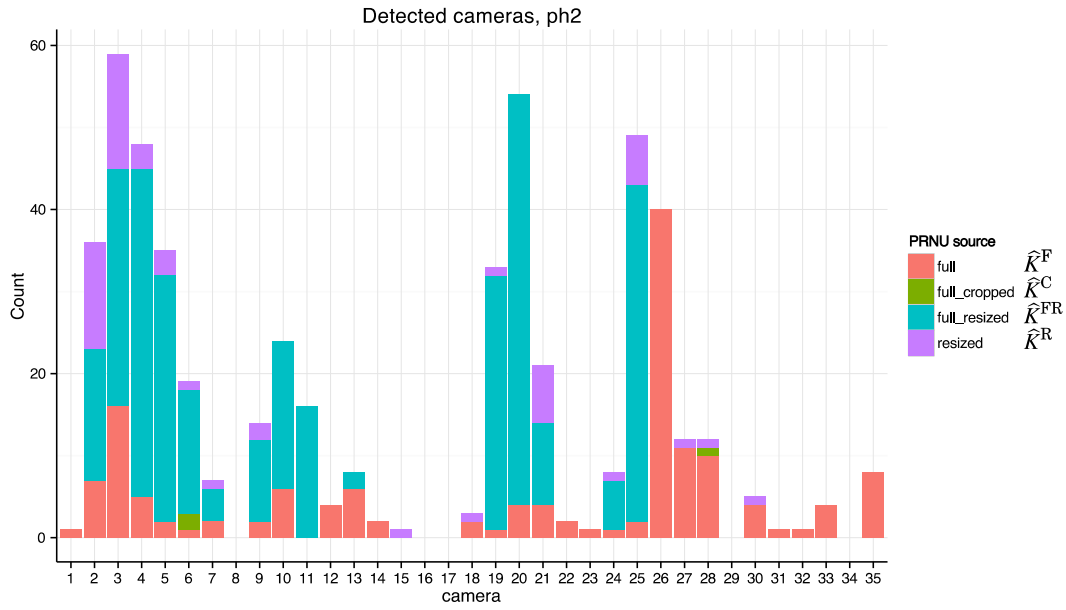
In particular, conditions that favor a blind clustering are found into the top-right part of each plot: that is, large amounts of images taken by the same camera, whose attribution is safe to be assumed (i.e. higher median PCE with lower variance). Judging by the relatively large amount of clusters of well-identified images, we conclude that blind clustering is possible, although not exactly favorable.

Resize

Here, we discuss whether $\widehat{K}^{\text{FR}}$ can be estimated with $\widehat{K}^{\text{R}}$, the sole PRNU fingerprint which can be actually computed from a fully blind standpoint, such as the Challenge.

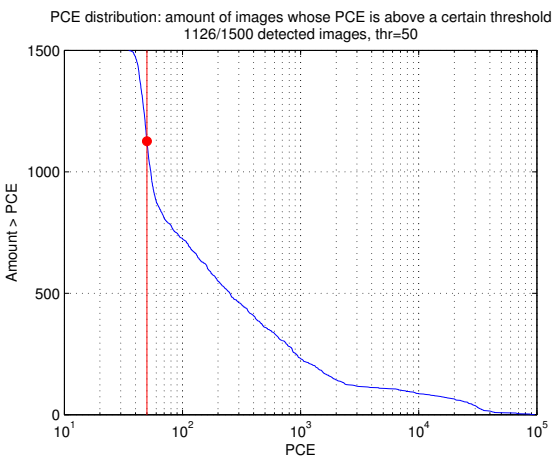


(a) Images in $\mathcal{I}_{\mathtt{tr}}$. x-axis: k-th SAD camera.

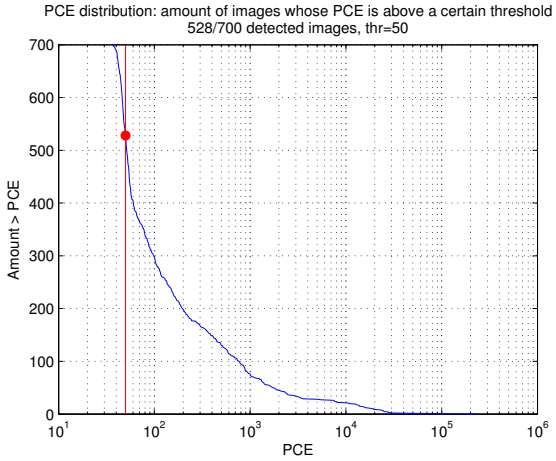


(b) Images in $\mathcal{I}^2_{\mathtt{ts}}$. x-axis: k-th SAD camera.

Figure 4.21: Distribution of attributed images for each camera in the SAD in each dataset across $\mathscr{K}_n$. The total height of each bar is the total number of images which have been attributed (i.e. the PCE between the image noise and the camera fingerprint is above $\tau = 50$) to the chosen camera. Notice how some cameras have never been used in the Challenge dataset.



(a) PCE distribution with attributed camera in $\mathcal{I}_{\mathsf{tr}}$



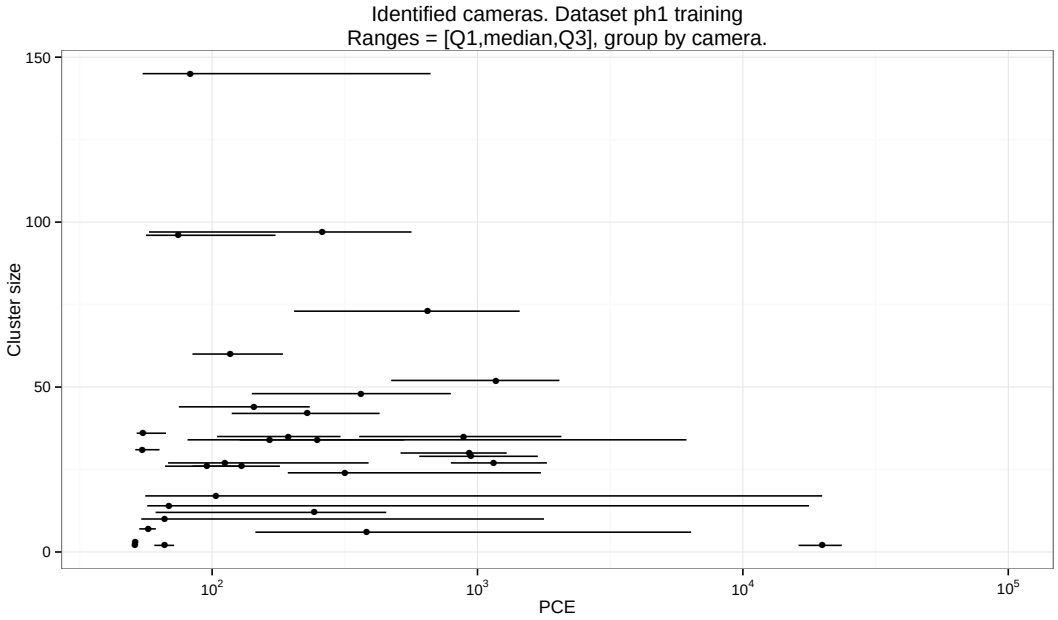
(b) PCE distribution with attributed camera in $\mathcal{I}^2_{\mathsf{ts}}$

Figure 4.22: Distribution of the best PCE obtained for each image against all SAD cameras using all $\mathscr{K}_n$. The red mark is the PCE detection threshold $\tau = 50$. These plots show how many images can be matched with the real camera over a given PCE.

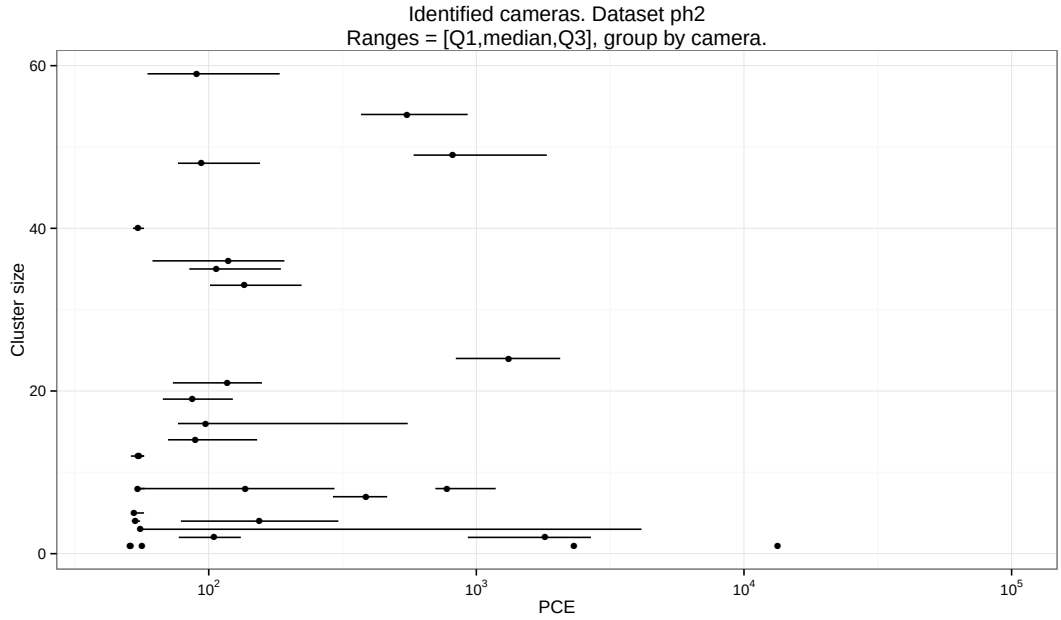
In the parallel coordinates plot in Figure 4.25, we show the PCE values of *all* images that have been detected using $\widehat{K}^{\mathrm{FR}}$, $\widehat{K}^{\mathrm{R}}$ and, for comparison purposes, $\widehat{K}^{\mathrm{F}}$. Instead of plotting cameras rather than groups of images, here we represent *each* image as a 4-point polygonal line, whose ordinates are the said PCE values against each kind of PRNU, arbitrarily and equispatially represented on the absissae. The advantage of a parallel coordinates plot is that it shows complex relationships between categorical variables, such as the four PRNUs, coupled with as many continuous responses such as the PCE values. The ordering of the absissae is actually relevant only for representation purposes: the vertices constituting the polygonal line can be swapped and moved in order to uncover more patterns.

It is clear from subfigures (a), (b), (c) and (d) how images that match with $\widehat{K}^{\mathrm{FR}}$ also match using $\widehat{K}^{\mathrm{R}}$, and vice versa, without *ever* matching also with $\widehat{K}^{\mathrm{F}}$. Moreover, observe (subfigures (e) and (f)) how images that match with $\widehat{K}^{\mathrm{C}}$ also match with $\widehat{K}^{\mathrm{F}}$: this is due to the fact that only a cropping has been performed between the two PRNU. This allows us to conclude that Hyp. II is not strictly necessary for camera attribution, albeit a size correction on PCE is needed. Furthermore, as we can note from subfigures (e) and (f), the same does *not* happen with $\widehat{K}^{\mathrm{F}}$ against $\widehat{K}^{\mathrm{FR}}$ and $\widehat{K}^{\mathrm{R}}$, since a scaling is now involved.

As an additional proof, in Table 4.7 we reported how many times the same camera is attributed in $\mathcal{I}^2_{\mathsf{ts}}$ either using one kind of PRNU or another. Granted the preceding observations, we can appreciate that, very often, $\widehat{K}^{\mathrm{FR}}$ and $\widehat{K}^{\mathrm{R}}$

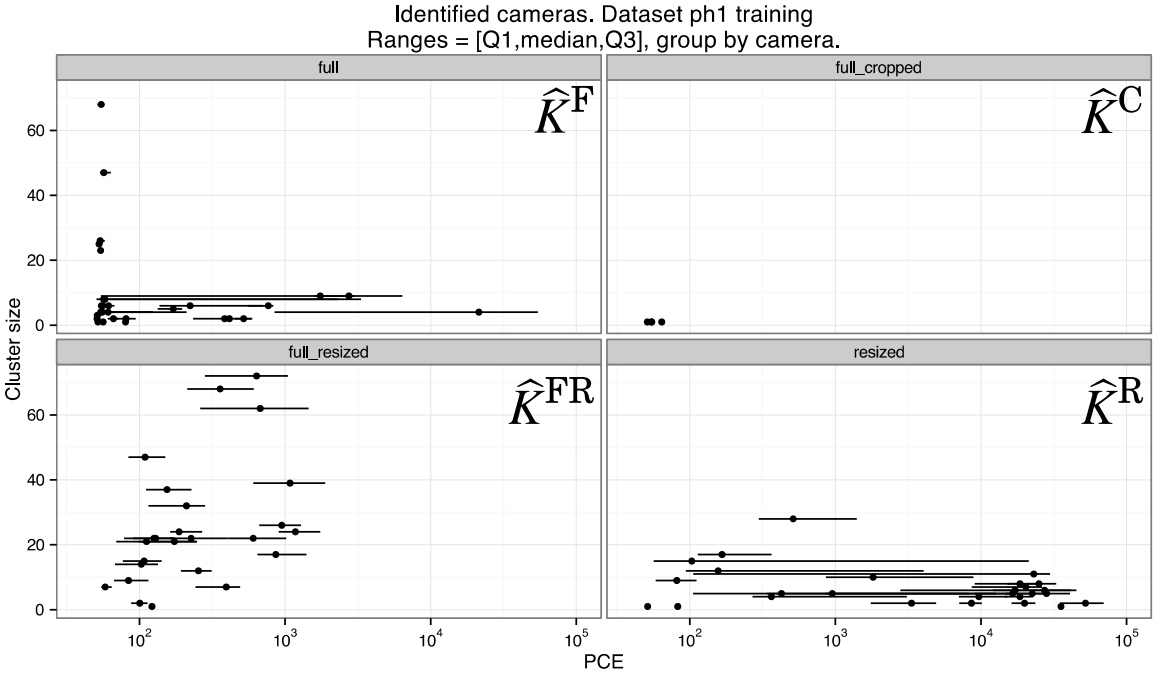


(a) Images in $\mathcal{I}_{\mathtt{tr}}$, with PCE $> \tau = 50$

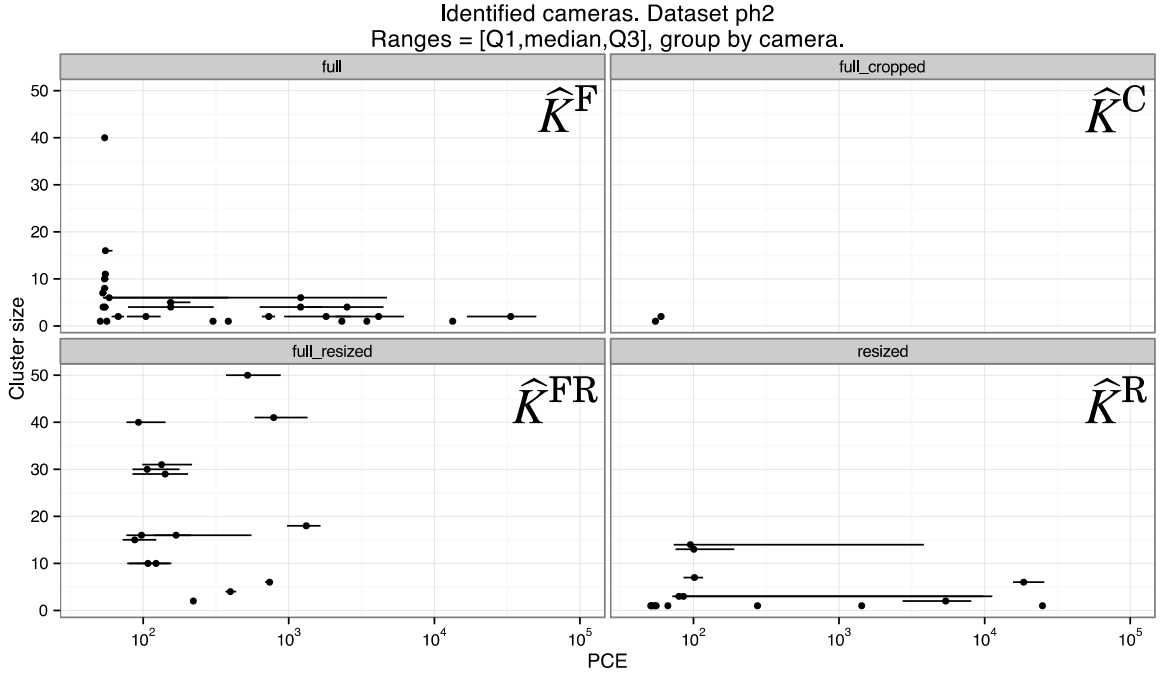


(b) Images in $\mathcal{I}^2_{\mathtt{ts}}$, with PCE $> \tau = 50$

Figure 4.23: Detectability plot. Each horizontal bar is a cluster of images taken by the same SAD camera, as recognized by at least one of the $\mathscr{K}$. Non-matched images are not shown.
The cluster size is the y-coordinate, while on the x-axis we reported the median, first and third quartile of the PCE value in each cluster against its SAD camera.
Ideal conditions for blind clustering can be found in the upper right part of this graph: that is, images are easy to identify, and each camera is represented by lots of pictures. Also, a lower variance means that the influence on image content is negligible: e.g. images have not been aggressively compressed, or any alteration has a small area.



(a) Images in $\mathcal{I}_{\mathtt{tr}}$, with PCE $> \tau = 50$



(b) Images in $\mathcal{I}^2_{\mathtt{ts}}$, with PCE $> \tau = 50$

Figure 4.24: Same plot as Figure 4.23: this time, however, we group each cluster along each one of the PRNU types $\mathscr{K}$.

attribute the same camera to each identifiable image: again, in other cases this does happen much more rarely, e.g. between $\widehat{K}^{\mathrm{F}}$ and $\widehat{K}^{\mathrm{R}}$.

It is thus immediate to conclude that the operation of resizing to CR ($\mathscr{R}$) can be safely swapped with the PRNU extraction.

Attributed by	also matching with			
	$\widehat{K}^{\mathrm{F}}$	$\widehat{K}^{\mathrm{C}}$	$\widehat{K}^{\mathrm{FR}}$	$\widehat{K}^{\mathrm{R}}$
$\widehat{K}^{\mathrm{F}}$	150	22	8	6
$\widehat{K}^{\mathrm{C}}$	0	3	0	0
$\widehat{K}^{\mathrm{FR}}$	5	9	318	303
$\widehat{K}^{\mathrm{R}}$	1	0	48	57

Table 4.7: Camera attributions in $\mathcal{I}^2_{\mathsf{ts}}$ between different sources. In (i, j)-th entry we reported the amount of images whose attributed camera is the same using i-th PRNU type against j-th PRNU type. As a result, one can read on the diagonal how many images are identified using i-th PRNU type.
E.g. in $\mathcal{I}^2_{\mathsf{ts}}$ we were able to assign a camera to 150 images using $\widehat{K}^{\mathrm{F}}$. Among these images, in 22 cases we would not attributed another camera if we used $\widehat{K}^{\mathrm{C}}$ instead.

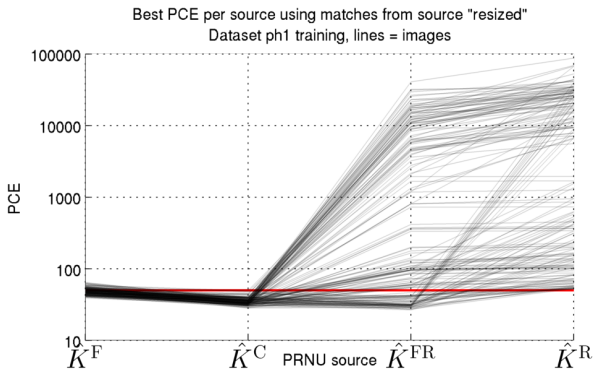
Comparison with blind results

To further verify our assertions, we compared cameras as attributed using the SAD with the blind clustering described in 4.1.2.

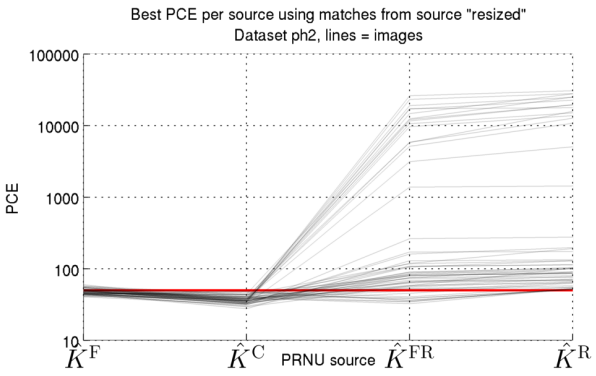
In Figure 4.26 we represented the PCE obtained by correlating noise extracted from an image $I \in \mathcal{I}_{\mathsf{tr},\mathsf{F}}$, set as fixed, with noises extracted from the entire dataset $\mathcal{I}_{\mathsf{tr}} \cup \mathcal{I}^2_{\mathsf{ts}}$. Only 1024×768 pixel wide images are considered. Superimposed, we also marked images whose camera has been attributed to be the same who took I, by the two approaches.

From subfigure (a), one can see how the blind procedure is much more restrictive than using prior information (i.e. the SAD): in fact, blind clusters are almost always subsets of non-blind clusters. Moreover, it (almost) *never* declares that two images have been taken by different cameras, given that the camera who took them is actually the same (as attributed by the SAD).

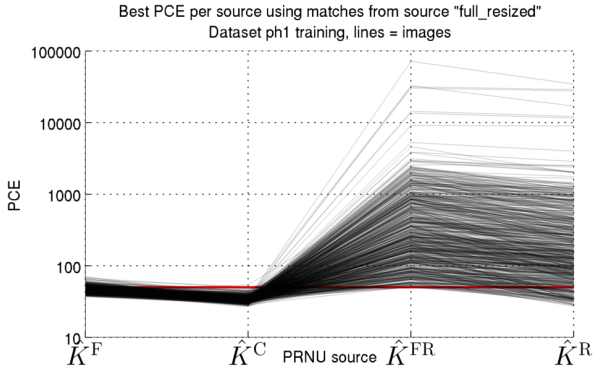
Another remark is the presence of near-duplicates inside the dataset (see subfigure (b)): as the area in common between them is usually large, their mutual PCE is much higher than any other image. As a consequence, they are forcibly clustered together by the blind PCE-based clustering algorithm. In principle, this is not wrong; however, in near-duplicate-dominated clusters, we conjecture that the PRNU approach becomes similar to a phylogenetic



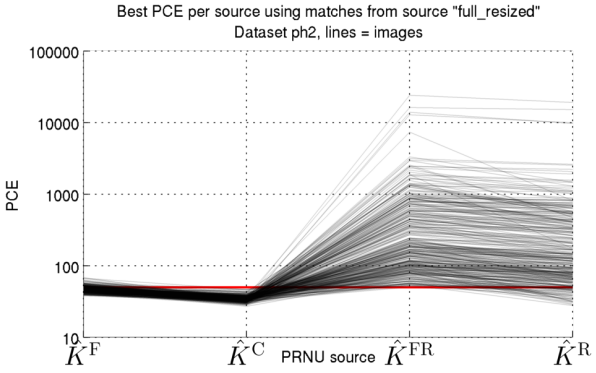
(a) PCE values of images in $\mathcal{I}_{\mathsf{tr}}$ whose camera has been attributed using $\widehat{K}^{\mathrm{R}}$.



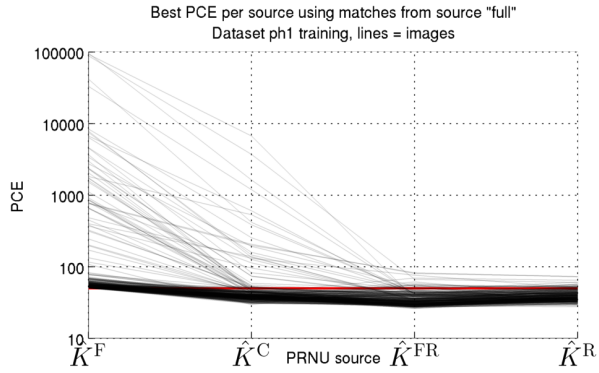
(b) PCE values of images in $\mathcal{I}^2_{\mathsf{ts}}$ whose camera has been attributed using $\widehat{K}^{\mathrm{R}}$.



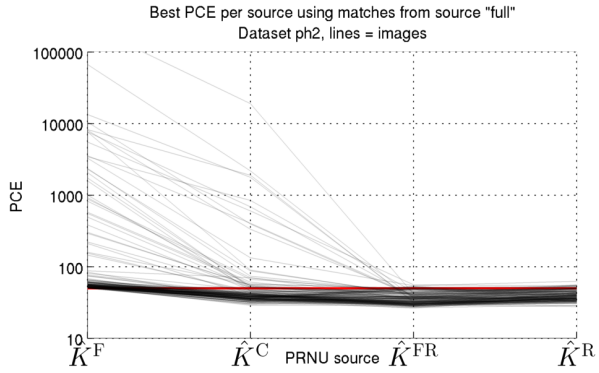
(c) PCE values of images in $\mathcal{I}_{\mathsf{tr}}$ whose camera has been attributed using $\widehat{K}^{\mathrm{FR}}$.



(d) PCE values of images in $\mathcal{I}^2_{\mathsf{ts}}$ whose camera has been attributed using $\widehat{K}^{\mathrm{FR}}$.



(e) PCE values of images in $\mathcal{I}_{\mathsf{tr}}$ whose camera has been attributed using $\widehat{K}^{\mathrm{F}}$.



(f) PCE values of images in $\mathcal{I}^2_{\mathsf{ts}}$ whose camera has been attributed using $\widehat{K}^{\mathrm{F}}$.

Figure 4.25: Distribution of PCE values of all *images whose camera has been attributed using at least one of the $\mathscr{K}$. Each image is represented as a line. The red mark is the PCE threshold $\tau = 50$.*
Columns: images in $\mathcal{I}_{\mathsf{tr}}$ and $\mathcal{I}^2_{\mathsf{ts}}$ respectively.
Rows: images attributed using $\widehat{K}^{\mathrm{R}}$, $\widehat{K}^{\mathrm{FR}}$ and $\widehat{K}^{\mathrm{F}}$ respectively.
E.g. an image in $\mathcal{I}_{\mathsf{tr}}$ which is matched using $\widehat{K}^{\mathrm{F}}$ would, probably, have a peak in the first column in (e) and (f) well above τ; in other plots and columns, instead, the reported PCE values would be well under τ.

approach using such near duplicates.

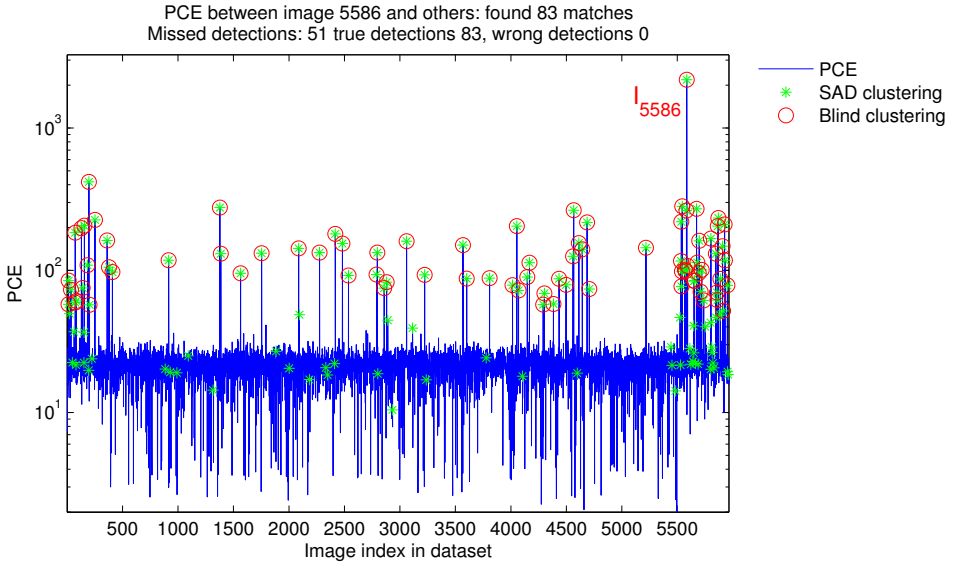
4.2.7 Feature-based approach

Notwithstanding what we concluded in Section 3.4 about co-occurrence-based method for tampering detection, we also tried to adapt the same for tamper localization purposes.

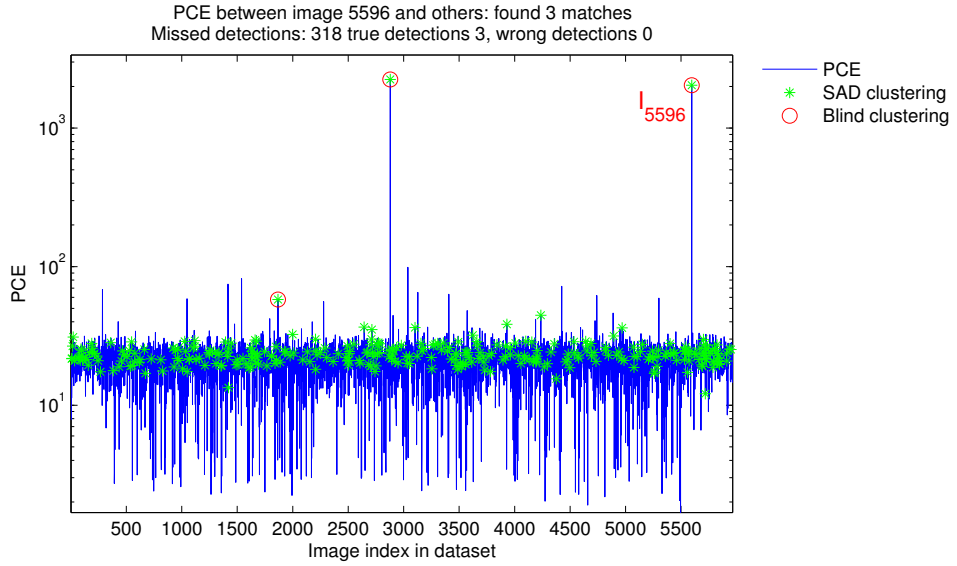
Following what we did in 3.3.2, we collectively extracted 16678 64×64 pixels wide non-overlapping blocks from all images in $\mathcal{I}_{\mathsf{tr},\mathsf{F}}$. The key idea can also be found in the winning approach described in 2.3.2: to each block we associated the class label l_i, set to 0 (i.e. *pristine* block) if its mask is either completely white or completely black, while l_i is set to 1 (i.e. *fake* block) if the block is composed from 25% to 75% black pixels. We discarded blocks which did not comply to any of these requirements. As a result, we basically extracted blocks either fully contained into a black (or white) region, or blocks across the edges of any tampering.

Next, we trained the SVM classifiers on the obtained dataset, with the same cross-validation setup as before.

First obtained results were not encouraging at all, even in training phase, especially if compared with those obtained using entire images in the Detection phase. (see Figure 4.27a against Figure 4.27b). Even a change of block size or feature normalization did not help: as a result, we decided not to consider this approach for the Localization phase.

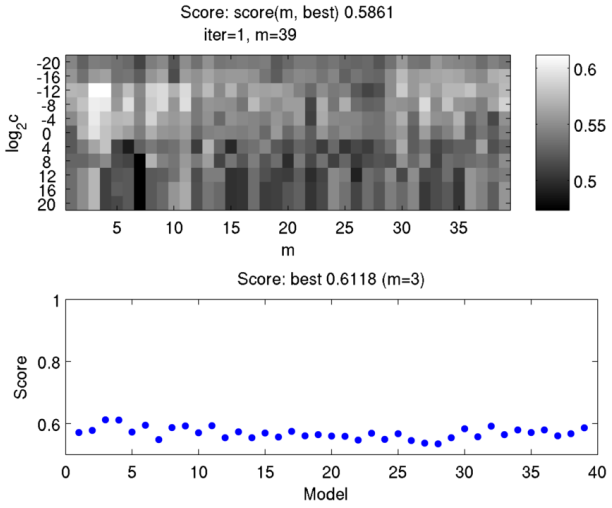


(a) Reference image I_{5586}.

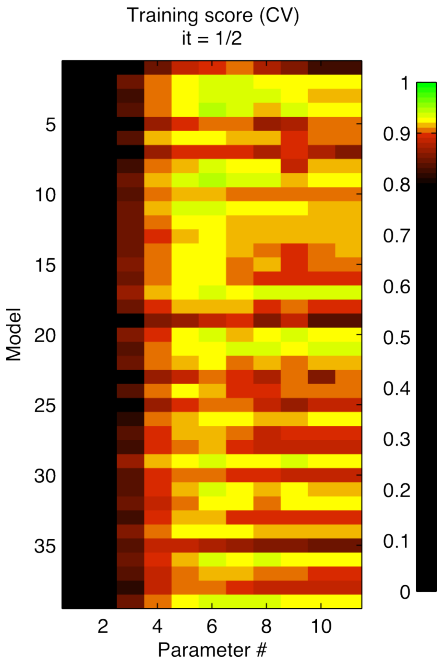


(b) Reference image I_{5596}.

Figure 4.26: SAD/Blind clustering comparison. We fixed image $I_n \in \mathcal{I}_{\mathtt{tr}} \cup \mathcal{I}^2_{\mathtt{ts}}$. Images marked with $*$ have been attributed to the same camera as I_n using SAD. Red circles, instead, mark images attributed to the same camera as I_n using our blind clustering algorithm.



(a) Grid search results using the co-occurrence-based approach. Only one grid refinement is performed. In the lower graph, we report the best score reached by each submodel.
Notice also that we are inside the training set, therefore expected score on test set is lower.



(b) Grid search score obtained in the training phase of the tampering global detector, presented in Section 3.1.
Color: score, y-axis: submodel m, x-axis: index of the chosen metaparameter inside m-th metaparameter grid (same as the one used in (a)).

Figure 4.27: Grid search scores obtained in the feature-based detector adapted for tamper localization (a). For clarity, in (b) we reported the scores obtained by the same feature-based detector in the Detection phase, using full images instead, already shown in Figure 3.2a.

4.3 Conclusions

In this chapter we presented three techniques, each one bearing different strenghts and weaknesses. We also showed a possible way to combine said techniques in order to produce a single decision for each image.

In particular, we first exposed our novel PRNU-based approach. Compared to thresholding-based methods, as we verified in Subsection 4.2.2, our method requires *no* training at all except for the choice of the block partitioning. Reported scores using M_{PRNU} are still on-par with PCE-based methods, both on the training set $\mathcal{I}_{\text{tr,F}}$ and on the test set $\mathcal{I}^2_{\text{ts}}$: however, we note that we mostly analyzed the performance of M_{PCE} on the same dataset used for training it. We therefore expect that, in general, M_{PCE} performs worse than M_{PRNU} on entirely new images.

The Near-Duplicate based technique we presented is borrowed from the image phylogeny field. The possibility of having multiple versions of a single image is a rather strong requirement, evidenced by the fact that we were able to find only 100 near-duplicates over 700 in the Challenge phase 2 dataset. Nevertheless, image phylogeny is a relatively new field to work on, and the performance exhibited by M_{ND} (when available) is much encouraging to further work with the individual method.

To conclude the chapter, we have proven how any forensic tool, alone, is not able to reach an acceptable accuracy and detection rate. We have also shown that the right path is to merge the outputs of many forensic tools, possibly in a way both to strenghten each other and to replace a technique which fails on the analyzed image.

As far as the Challenge results, we achieved the score of $\mathbf{0.453273}$, well over the score obtained by the winning submission (0.407172). As the scoring mechanism involves a certain randomness, the superiority is further confirmed by multiple submissions of the masks produced by the proposed technique: all of them were well over the winning score.

Chapter 5

Conclusions

In this work we tackled two classic problems in image forensics: tampering detection and tampering localization. In both cases, we proposed a method built on current state-of-the-art techniques: we showed that suggested methods achieve better results in a fully blind fashion on the Challenge dataset than those who won the IEEE IFS-TC Image Forensic Challenge.

In particular, in the Detection phase we refined the approach followed by [45], showing that further improvement is possible through a more advanced ensemble classifier, such as a Boosted Logistic Regression.

In the Localization phase, instead, we proposed an approach which combines our novel take on PRNU-based forensic methods, with tools coming from the image phylogeny world and a technique capable of revealing copy-move.

The developed PRNU-based method requires no training as opposed to other available techniques, but it is capable of achieving the same score on the Challenge dataset. We also proved the validity of our approach by comparison with another dataset, the SAD, giving conditions for the establishment of a fully blind clustering based on cameras which have been used to produce the dataset.

The Near-Duplicate approach, instead, relies on a redundancy exhibited by this dataset. Having said that, the shown technique is very powerful whenever a set of near-duplicates can be identified: in fact, in most cases we are able to exactly reconstruct the ground truth mask by comparing pairs of images. Also, Near-Duplicate analysis is performed very quickly and reliably. The major drawback is that it is prone to failure each time a tampering occurs in each image. Still, this technique can be employed if the analyzed dataset is very large, as it increases the possibility to locate similar images to the one which is being questioned: for example, this can be exploited by crawling the Web for near-duplicate images, as tools such as Tineye or Google Images are

capable of doing. Being also very light on computational resources, the fertile ground for this technique is a distributed computing platform: this enables the analyst to exploit a collection which is too large to be stored locally, in order to find near-duplicates extraneous to the examined collection.

However, a number of questions are left open.

In the Detection phase, we proved that the proposed detector has learned to classify features that characterize entire images rather than local alterations: however, those features do not necessarily suggest that a tampering has been performed in the image. Their nature is still unknown: we conjecture that the detector is able to recognize whether a re-save or a color space conversion has taken place, rather than the editing program which has been used to author the image. Future work is aimed in this direction: in particular, we wish to investigate which one is the aspect that it is actually detected. To do so, we will produce a series of datasets starting from pristine images in $\mathcal{I}_{\mathtt{tr}}$: each dataset is processed so to enhance differences introduced by each kind of said global processing. To be able to detect such features is a factor likable for further forensic work.

In the Localization phase, the proposed PRNU-based method is strongly dependent on the choice of block partitioning which is performed on the image. We dealt with this aspect by a rather crude grid search on the space of block parameters. It would be desirable to have a lower bound on the block size, or to be able to define a link between block parameters and the performance of the method: this could further decrease the training required by its implementation. To make sure that the blind PRNU clustering is well posed, we also assumed both that no cropping and no resize has been performed on the images and that they could have only been resized to a resolution of 1024×768 pixels. We showed that these requirements can be relaxed, verifying it experimentally. Further work could be oriented in this direction: in particular, it would be desirable to allow a wider spectrum of operators which could be applied to analyzed images, such as resizing to arbitrary resolution or multiple re-compression.

Near-Duplicate analysis is very promising due to the high scores which can be achieved by its use. In this work we described a way to synthesize a single mask by comparing the analyzed images with a set of near-duplicates: we proved that, in some cases, this does not lead to a correct decision. However, this could be circumvented by comparing tuples of near-duplicates, instead of pairs. A weakness shared both by Near-Duplicate analysis and PatchMatch is that they exhibit ambiguities: we proposed a way to disambiguate the obtained masks by collectively merging different techniques. Many other ways are possible: in particular, one could also exploit the fact that some of

the near-duplicates can also actually be pristine. Another improvement would be to perform mask disambiguation using PRNU-based information.

A further development on the Near-Duplicate idea is to refine the underlying measure of similarity between images: in our approach we compare images pixel by pixel. However, among the near-duplicates we may allow for slight changes in the represented scene, perhaps to exclude small moving elements such as pedestrians, clouds, foliage, etc.; our approach, in fact, would classify them as being tampered regions. This leads us to rethink the definition we use for *fake*: untampered photographs of a real scene could be considered as being pristine, even though a depicted object could have moved between frames. As a result, one might shift the entire approach to detecting whether a given *scene* is fake. For example, suppose that an object has been subsequently added to a photo of a natural scene, whose a bunch of photographs are available, ranging both in time and in space. A Near-Duplicate analysis approach would, then, reconstruct the scene based on such photographs, and would detect the object as being erroneous by uncovering its inconsistency in time and space with the rest of the scene. As a consequence, the natural framework for this approach is video forensic analysis.

We finish this work by emphasizing the importance of fusing the outputs of several techniques in order to produce a single decision: this has been the staple both for Phase 1 (i.e. the ensemble classifier) and Phase 2 (mask fusion). In the latter we employed a rather simple way to synthesize a single mask for each image. Further work could be spent to refine this process. An idea is to merge existing prior information, both on the analyzed image and on each mask: the former stems from the knowledge of statistics exhibited by natural images, while the latter is proper to the employed techniques. To this purpose, for example, one might employ tools borrowed from Bayesian statistics. In forensic literature a similar approach has been done in [56].

All of this proves that image forensics is a field where significant improvement over state-of-the-art methods is still possible, also benefiting from entirely novel ideas coming from unrelated fields.

Bibliography

[1] A. Piva, "An Overview on Image Forensics", *ISRN Signal Processing*, vol. 2013, pp. 1–22, 2013, ISSN: 2090-505X. DOI: 10.1155/2013/496701.

[2] S. Milani, M. Fontani, P. Bestagini, M. Barni, A. Piva, M. Tagliasacchi, and S. Tubaro, "An overview on video forensics", *APSIPA Transactions on Signal and Information Processing*, vol. 1, Aug. 2012, ISSN: 2048-7703. DOI: 10.1017/ATSIP.2012.2.

[3] I. J. Cox, M. L. Miller, J. A. Bloom, and C. Honsinger, *Digital watermarking*. Springer, 2002, vol. 53.

[4] S. Gupta, S. Cho, and C.-C. Kuo, "Current developments and future trends in audio authentication", *MultiMedia, IEEE*, vol. 19, no. 1, pp. 50–59, 2012.

[5] M. Visentini-Scarzanella and P. L. Dragotti, "Modelling radial distortion chains for video recapture detection", in *Multimedia Signal Processing (MMSP), 2013 IEEE 15th International Workshop on*, IEEE, 2013, pp. 412–417.

[6] M. K. Johnson and H. Farid, "Exposing digital forgeries through chromatic aberration", in *Proceedings of the 8th workshop on Multimedia and security*, ACM, 2006, pp. 48–55.

[7] L. T. Van, S. Emmanuel, and M. S. Kankanhalli, "Identifying source cell phone using chromatic aberration", in *Multimedia and Expo, 2007 IEEE International Conference on*, IEEE, 2007, pp. 883–886.

[8] I. Yerushalmy and H. Hel-Or, "Digital image forgery detection based on lens and sensor aberration", *International journal of computer vision*, vol. 92, no. 1, pp. 71–91, 2011.

[9] A. E. Dirik, H. T. Sencar, and N. Memon, "Digital single lens reflex camera identification from traces of sensor dust", *Information Forensics and Security, IEEE Transactions on*, vol. 3, no. 3, pp. 539–552, 2008, ISSN: 1556-6013.

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

[11] A. Swaminathan, M. Wu, and K. R. Liu, "Digital image forensics via intrinsic fingerprints", *Information Forensics and Security, IEEE Transactions on*, vol. 3, no. 1, pp. 101–117, 2008, ISSN: 1556-6013.

[12] A. C. Popescu and H. Farid, "Exposing digital forgeries in color filter array interpolated images", *Signal Processing, IEEE Transactions on*, vol. 53, no. 10, pp. 3948–3959, 2005, ISSN: 1053-587X.

[13] P. Ferrara, T. Bianchi, A. De Rosa, and A. Piva, "Image forgery localization via fine-grained analysis of cfa artifacts", *Information Forensics and Security, IEEE Transactions on*, vol. 7, no. 5, pp. 1566–1577, 2012, ISSN: 1556-6013.

[14] M. Chen, J. Fridrich, M. Goljan, and J. Lukas, "Determining Image Origin and Integrity Using Sensor Noise", *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 1, pp. 74–90, 2008, ISSN: 1556-6013. DOI: 10.1109/TIFS.2007.916285.

[15] M. Goljan and J. Fridrich, "Sensor-fingerprint based identification of images corrected for lens distortion", in *IS&T/SPIE Electronic Imaging*, International Society for Optics and Photonics, 2012, 83030H–83030H.

[16] ——, "Camera identification from cropped and scaled images", in *Electronic Imaging 2008*, International Society for Optics and Photonics, 2008, 68190E–68190E.

[17] C. McKay, A. Swaminathan, H. Gou, and M. Wu, "Image acquisition forensics: Forensic analysis to identify imaging source", in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, IEEE, 2008, pp. 1657–1660.

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

[19] Z. Fan and R. L. de Queiroz, "Identification of bitmap compression history: JPEG detection and quantizer estimation", *Image Processing, IEEE Transactions on*, vol. 12, no. 2, pp. 230–235, 2003.

[20] D. Fu, Y. Q. Shi, and W. Su, "A generalized Benford's law for JPEG coefficients and its applications in image forensics", in *Electronic Imaging 2007*, International Society for Optics and Photonics, 2007, pp. 65051L–65051L.

[21] T. Bianchi and A. Piva, "Detection of non-aligned double JPEG compression with estimation of primary compression parameters", in *Image Processing (ICIP), 2011 18th IEEE International Conference on*, IEEE, 2011, pp. 1929–1932.

[22] A. J. Fridrich, B. D. Soukal, and A. J. Lukás, "Detection of copy-move forgery in digital images", in *in Proceedings of Digital Forensic Research Workshop*, Citeseer, 2003.

[23] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing", *ACM Transactions on Graphics (Proc. SIGGRAPH)*, vol. 28, no. 3, Aug. 2009.

[24] I. Amerini, L. Ballan, R. Caldelli, A. Del Bimbo, and G. Serra, "Geometric tampering estimation by means of a SIFT-based forensic analysis", in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, IEEE, 2010, pp. 1702–1705.

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

[26] B. Mahdian and S. Saic, "Blind authentication using periodic properties of interpolation", *Information Forensics and Security, IEEE Transactions on*, vol. 3, no. 3, pp. 529–538, 2008.

[27] M. K. Johnson and H. Farid, "Exposing digital forgeries by detecting inconsistencies in lighting", in *Proceedings of the 7th workshop on Multimedia and security*, ACM, 2005, pp. 1–10.

[28] M. K. Johnson and H. Farid, "Exposing digital forgeries in complex lighting environments", *Information Forensics and Security, IEEE Transactions on*, vol. 2, no. 3, pp. 450–461, 2007.

[29] C. Riess and E. Angelopoulou, "Scene illumination as an indicator of image manipulation", in *Information Hiding*, Springer, 2010, pp. 66–80.

[30] W. Zhang, X. Cao, J. Zhang, J. Zhu, and P. Wang, "Detecting photographic composites using shadows", in *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on*, Jun. 2009, pp. 1042–1045. DOI: 10.1109/ICME.2009.5202676.

[31] Q. Liu, X. Cao, C. Deng, and X. Guo, "Identifying image composites through shadow matte consistency", *Information Forensics and Security, IEEE Transactions on*, vol. 6, no. 3, pp. 1111–1122, 2011.

[32] M. K. Johnson and H. Farid, "Detecting photographic composites of people", in *Digital Watermarking*. Springer, 2008, pp. 19–33.

[33] V. Conotter, G. Boato, and H. Farid, "Detecting photo manipulation on signs and billboards", in *Image Processing (ICIP), 2010 17th IEEE International Conference on*, IEEE, 2010, pp. 1741–1744.

[34] H. Farid and J. Kosecka, "Estimating planar surface orientation using bispectral analysis", *Image Processing, IEEE Transactions on*, vol. 16, no. 8, pp. 2154–2160, 2007.

[35] H. Yu, T.-T. Ng, and Q. Sun, "Recaptured photo detection using specularity distribution", in *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, IEEE, 2008, pp. 3140–3143.

[36] X. Gao, T.-T. Ng, B. Qiu, and S.-F. Chang, "Single-view recaptured image detection based on physics-based features", in *Multimedia and Expo (ICME), 2010 IEEE International Conference on*, IEEE, 2010, pp. 1469–1474.

[37] J. Lukas, J. Fridrich, and M. Goljan, "Digital camera identification from sensor pattern noise", *Information Forensics and Security, IEEE Transactions on*, vol. 1, no. 2, pp. 205–214, 2006.

[38] I. Amerini, R. Caldelli, V. Cappellini, F. Picchioni, and A. Piva, "Analysis of denoising filters for photo response non uniformity noise extraction in source camera identification", in *Digital Signal Processing, 2009 16th International Conference on*, IEEE, 2009, pp. 1–7.

[39] G. Chierchia, S. Parrilli, G. Poggi, C. Sansone, and L. Verdoliva, "On the influence of denoising in PRNU based forgery detection", in *Proceedings of the 2nd ACM workshop on Multimedia in forensics, security and intelligence*, ACM, 2010, pp. 117–122.

[40] M. K. Mıhçak, I. Kozintsev, and K. Ramchandran, "Spatially adaptive statistical modeling of wavelet image coefficients and its application to denoising", presented at the Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on, vol. 6, IEEE, 1999, pp. 3253–3256, ISBN: 0780350413.

[41] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering", *Image Processing, IEEE Transactions on*, vol. 16, no. 8, pp. 2080–2095, 2007.

[42] T. Gloe, S. Pfennig, and M. Kirchner, "Unexpected artefacts in PRNU-based camera identification: a'Dresden Image Database'case-study", in *Proceedings of the on Multimedia and security*, ACM, 2012, pp. 109–114.

[43] M. Goljan, J. Fridrich, and J. Lukao, "Camera Identification from Printed Images", in *Proceedings of SPIE, the International Society for Optical Engineering*, Society of Photo-Optical Instrumentation Engineers, 2008, pp. 68190I–1.

[44] G. Chierchia, G. Poggi, C. Sansone, and L. Verdoliva, "PRNU-based forgery detection with regularity constraints and global optimization", in *Multimedia Signal Processing (MMSP), 2013 IEEE 15th International Workshop on*, IEEE, 2013, pp. 236–241.

[45] D. Cozzolino, D. Gragnaniello, and L. Verdoliva, "Image forgery detection based on the fusion of machine learning and block-matching methods", *arXiv preprint arXiv:1311.6934*, 2013.

[46] J. Fridrich and J. Kodovsky, "Rich models for steganalysis of digital images", *Information forensics and security, IEEE transactions on*, vol. 7, no. 3, pp. 868–882, 2012.

[47] D. Cozzolino, D. Gragnaniello, and L. Verdoliva, "A novel framework for image forgery localization", *arXiv preprint arXiv:1311.6932*, 2013.

[48] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines", *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, 27:1–27:27, 2011.

[49] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A Library for Large Linear Classification", *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[50] T. M. Cover, "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition", *Electronic Computers, IEEE Transactions on*, no. 3, pp. 326–334, 1965.

[51] L. Gaborini, P. Bestagini, S. Milani, Marco Tagliasacchi, and S. Tubaro, "Multi-Clue Image Tampering Localization", in *Workshop on Information Forensics and Security 2014 (GlobalSIP14-Workshop on Information Forensics and Security 2014)*, Atlanta, USA, Dec. 2014.

[52] S. Lameri, P. Bestagini, A. Melloni, S. Milani, A. Rocha, M. Tagliasacchi, and S. Tubaro, "Who is my parent? Reconstructing video sequences from partially matching shots", in *IEEE International Conference on Image Processing (ICIP)*, 2014.

[53] Z. Dias, A. Rocha, and S. Goldenstein, "Image Phylogeny by Minimal Spanning Trees", *IEEE Transactions on Information Forensics and Security*, vol. 7, pp. 774–788, 2012, ISSN: 1556-6013. DOI: 10.1109/TIFS.2011.2169959.

[54] A. De Rosa, F. Uccheddu, A. Piva, M. Barni, and A. Costanzo, "Exploring Image Dependencies: A New Challenge in Image Forensics", in *SPIE Conference on Media Forensics and Security (MFS)*, San Jose, CA, 2010.

[55] Z. Dias, A. Rocha, and S. Goldenstein, "First steps toward image phylogeny", in *IEEE International Workshop on Information Forensics and Security (WIFS)*, 2010. DOI: 10.1109/WIFS.2010.5711452.

[56] M. Fontani, E. Argones-Rua, C. Troncoso, and M. Barni, "The watchful forensic analyst: Multi-clue information fusion with background knowledge", in *Information Forensics and Security (WIFS), 2013 IEEE International Workshop on*, Nov. 2013, pp. 120–125. DOI: 10.1109/WIFS.2013.6707805.

[57] 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.

Appendices

Appendix A

Support Vector Machines

Support Vector Machines (SVM) is a set of supervised classifiers commonly employed in machine learning for classifying data [48]. Their general setup is the following. For further details, we recommend the seminal paper of Boser, Guyon, and Vapnik, [57].

Suppose that we have a set of N feature vectors, $\mathcal{S}_{\text{Tr}} \coloneqq \{\underline{x}_i\}_{i=1}^N$, each one of dimensionality p, i.e. $\underline{x}_i \in \mathbb{R}^p$. Being a supervised technique, we are also provided with a set of instance labels $\{l_i\}_{i=1}^N$, one for each data point. We restrict here to the case of *binary classification*: that is, $l_i \in -1, 1^p \quad \forall i \in 1, \dots, N$.

The goal of a SVM is to learn a relationship (i.e. *training*) between $\{\underline{x}_i\}_{i=1}^N$ and $\{l_i\}_{i=1}^N$, in order to be able to classify new data.

Specifically, suppose that we are given another set of features $\mathcal{S}_{\text{Ts}} \coloneqq \{\underline{x}_i\}_{i \in \mathcal{S}_{\text{Ts}}}$.

A SVM trained on $\mathcal{S}_{\text{Tr}}$ tries to predict unknown labels $\{l_i\}_{i \in \mathcal{S}_{\text{Ts}}}$ by applying the previously inferred relationship on $\mathcal{S}_{\text{Ts}}$.

To train a SVM requires the solution of the following optimization problem [57]

$$\begin{aligned} \min_{\underline{w}, b, \xi} \quad & \frac{1}{2} \underline{w}^\top \underline{w} + C \sum_{i=1}^N \xi_i \\ \text{s.t.} \quad & l_i \left(\underline{w}^\top \phi(\underline{x}_i) + b \right) \geq 1 - \xi_i \qquad \forall i \in 1, \dots, N \\ & \xi_i \geq 0 \end{aligned} \tag{A.1}$$

where $C > 0$ is a penalty parameter of the error term, ξ_i is related to the so-called *loss function* and ϕ is a function which maps input features $\underline{x}_i$ to a higher dimensional feature space, equipped with a dot product, in order to exploit the *kernel trick*.

In this context, a SVM finds the separating hyperplane with the maximum margin in such higher dimensional space.

The most important ingredient of this approach is the role of ϕ. In particular, instead of ϕ we may define the previous equation in terms of the so-called *kernel function*

$$K(\underline{x}_i, \underline{x}_j) \coloneqq \phi(\underline{x}_i)^\top \phi(\underline{x}_j). \tag{A.2}$$

The choice of K produces different support vector machines. In this work we use linear kernels, i.e. $\phi(x) = x$, therefore $K(\underline{x}_i, \underline{x}_j) = \underline{x}_i{}^\top \underline{x}_j$. The resulting SVM is, hence, called *linear SVM*. Notice that we are classifying the features in their own feature space.

Another common choice is the *Radial Basis Function*, i.e.

$$K(\underline{x}_i, \underline{x}_j) \coloneqq \exp\left(-\gamma \left\| \underline{x}_i - \underline{x}_j \right\|^2\right), \tag{A.3}$$

where $\|\cdot\|$ is the L^2 norm on $\mathbb{R}^p$.

In this work, SVM implementation is provided by the open-source library `LIBLINEAR` [49]. In particular, Equation A.1 can be written both in primal and dual form, with different choices of loss functions ξ. Further details can be found in `LIBLINEAR` manual.

Appendix B

The Challenge dataset

B.1 Scoring sensitivity

We investigated the sensitivity of the S_2 score used in Phase 2 of the Challenge. To this purpose, we simulated sets of Phase 2 submissions ($\mathcal{I}^2_{\mathsf{ts}}$) using fake images in the Phase 1 training set $\mathcal{I}_{\mathsf{tr,F}}$. We remind that $\left|\mathcal{I}_{\mathsf{tr,F}}\right| = 450$ while $\left|\mathcal{I}^2_{\mathsf{ts}}\right| = 700$.

Each set is characterized by a fundamental alteration which has been performed to the submitted masks. In most tests, we selected Phase 1 ground truth masks as starting masks to be modified: this is to evaluate the score sensitivity to small departures from the truth. Other tests, instead, were built to simulate random submissions, such as fully black masks or to evaluate the expected performance of a *fully random* classifier.

We also simulated the randomization mechanism to evaluate its impact on score variance, using the same randomization ratio reported in the Challenge rules.

We simulated the following tests:

- **White masks**: we scored 450 fully white masks.
- **Black masks**: we scored 450 fully black masks.
- **Salt & Pepper**: we scored 450 ground truth masks corrupted by salt & pepper noise, with $p = 0.05$ probability of corruption.
- **Random uniform**: we scored 450 masks whose pixels are sampled from a Bernoulli random variable of parameter $p = 0.5$. This is consistent with a fully random classifier.
- **Random negation**: we scored 450 ground truth masks whose pixels have been misclassified with probability $p \in \{0.05, 0.10, 0.20\}$.

- **Dilation**: we scored 450 ground truth masks whose white region has been enlarged by $P \in \{1,5,50\}$ pixels. This has been carried out with a disk shaped mathematical morphological operator, with P as its diameter.

- **Erosion**: we scored 450 ground truth masks whose black region has been enlarged by $P \in \{1,5,50\}$ pixels. This has been carried out with a disk shaped mathematical morphological operator, with P as its diameter.

Each test has been performed 20 times: we report the obtained results in Figure B.1.

Notice how small is the effect size of the randomization on the obtained score. Also, it is noteworthy how the S_2 score (i.e. F_1 score) favours black pixels over white ones (e.g. compare 5 pixel dilated masks with 5 pixel eroded masks).

Also, these scores are consistent with obtained Phase 2 results.

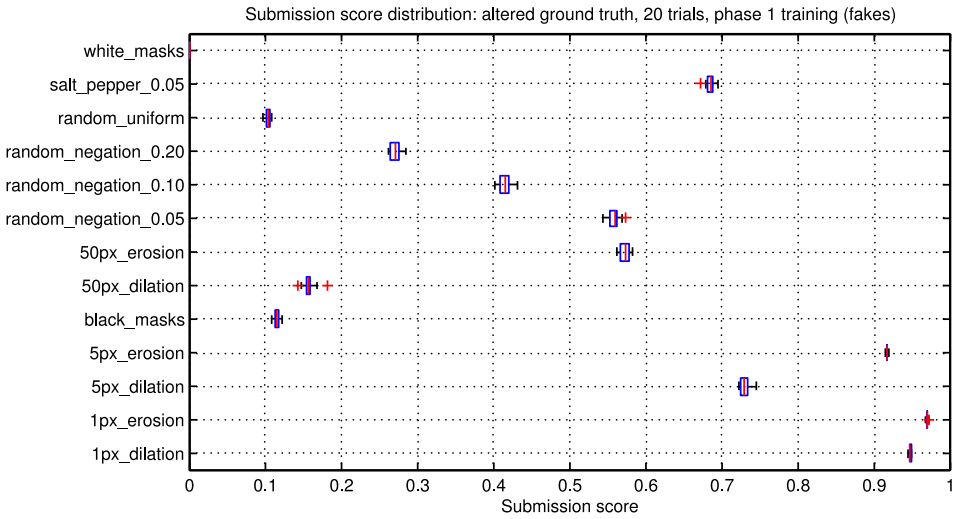


Figure B.1: Challenge score analysis on $\mathcal{I}_{\mathrm{tr,F}}$

To further test the scoring mechanism of the Challenge, we manually estimated all Phase 2 masks and submitted them. We were able to visually extract 347 masks out of 700: pairing with white masks, we obtained a score of 0.370564 (`submission_test_001` in Table B.2), well under the results achieved by the best algorithms.

B.2 Submission log

In Table B.1 we report the official scores obtained by all submissions we made, along with their details and compositions. We also made a second account solely for testing purposes (e.g. we submitted the said manually segmented masks): its obtained scores are reported, instead, in Table B.2.

Submission ID	Description	Score	Normalized score
submission_001	all zero (black) masks	0.121	
submission_002	all ones (white) masks	0	
submission_003	PM: masks obtained with PatchMatch only, the others are white, 371 masks	0.172293	0.325081
submission_004	PM: masks obtained with PatchMatch only, the others are black, 371 masks	0.22119	0.310925
submission_005	PM: same as submission_004.zip just to see the effect on the final score of random sampling the test set	0.214637	0.29856
submission_006	PM: masks obtained with dilated PatchMatch only, the others are black, 371 masks	0.27848	0.419019
submission_007	masks obtained with dilated PatchMatch intersected with manual annotations, the others are black, 371 masks	0.320341	0.498002
submission_008	FUS, PM-ND: submission_007 masks, substituting those obtained with NearDuplicates analysis	0.411923	
submission_009	FUS, PM-ND: PatchMatch from submission_006 intersected with automatically generated NearDuplicates masks (others are black)	0.397879	
submission_010	FUS, PM-ND: as submission_009	0.397359	
submission_011	masks from submission_007 intersected with automatically generated NearDuplicates masks (others are black)	0.406863	
submission_012	FUS, PM-ND: masks from NearDuplicates analysis, multi-way disambiguated using ND, no 2-way disambiguation, PM when no NDs are available, else black	0.425221	0.584467
submission_013	FUS, PM-ND: masks from NearDuplicates analysis, multi-way-disambiguated using ND, 2-way-disambiguated using PatchMatch, PM when no NDs are available, else black	0.404191	0.552465
submission_014	FUS, PM-ND: as submission_012	0.426302	
submission_015	ND: as submission_012, set to black if no ND are available	0.333147	0.844868
submission_016	PRNU: PRNU only, offset: 1 if no block offset is detected, w_size 64, w_step 8, 442 masks, 1024x768 images, others black	0.253417	0.331773
submission_017	ND: as submission_012, set to white if no ND are available	0.22133	0.351317
submission_018	FUS, ND-PM-PCE: submission_012 (ND), then submission_006 w/o black images (PM), then submission_test_006 (thresholded PCE), then black, 522 valid?	0.447132	0.558683
submission_019	FUS, ND-PM-PCE: submission_012 (ND), then submission_test_006 (thresholded PCE), then submission_006 w/o black images (PM), then black	0.415643	0.514943
submission_020	FUS, PRNU-PCE: AND between offset with interpolated PCE, thresholded at 4.0321, eroded with disk size 20, w_size 50, w_step 6, 442 masks, 1024x768 images, others white	0.237449	0.37605
submission_021	FUS, ND-PM-PRNU: submission_012 (ND), then PatchMatch_masks_eroded_strel_30_p2, then offset+morpho_wsz_64_wst_8_morpho_erosion_20_p2 (p1 strategy: 0.5559), then black, 506 masks	0.441684	0.565017
submission_022	submission_018	0.439578	0.548553
submission_023	submission_018	0.433462	0.540351
submission_024	submission_018	0.427923	0.532924
submission_025	submission_018	0.435124	0.54258
submission_026	submission_018	0.453273	0.566918

Table B.1: Challenge submissions using official user.

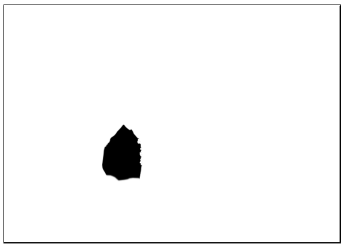
submission_test_001	masks manually segmented (white if nothing was manually found)	0.370564	
submission_test_002	masks manually segmented (black if nothing was manually found)	0.438558	
submission_test_003	masks automatically and manually (golden) obtained using near-duplicate analysis intersected with manually obtained ones (others black)	0.321625	
submission_test_004	as submission_003 using simple position-mapped PRNU masks when no PatchMatch/NearDuplicates are available	0.322053	
submission_test_006	PCE: interpolated PCE0, thresholded at 7, w_size 64, w_step 8, 441 masks, 1024x768 images, others white	0.262322	0.415442
submission_test_007	PRNU: as TeamGabor's submission_016: PRNU only, 1 if no block offset is detected, w_size 64, w_step 8, 442 masks, 1024x768 images, others white	0.2163	0.342557
submission_test_008	PCE: interpolated PCE0, thresholded at 4.0321, w_size 50, w_step 6, 441 masks, 1024x768 images, others white	0.24621	0.389925
submission_test_009	PCE: interpolated PCE0, thresholded at 4.0321, w_size 50, w_step 6, 441 masks, 1024x768 images, others white	0.262688	0.416022
submission_test_010	FUS, PCE-PRNU: AND between offset with interpolated PCE0, thresholded at 4.0321, w_size 50, w_step 6, 442 masks, 1024x768 images, others white	0.232247	0.367812
submission_test_013	PRNU: offset, offset+morpho_wsz_64_wst_8_morpho_erosion_20_p2, 1024x768 images, 364 masks, others white	0.238354	0.458373
submission_test_014	PM: PatchMatch_masks_eroded_strel_30_p2, others white	0.227011	0.468754

Table B.2: Challenge submissions using alternate user.

B.3 Dataset samples

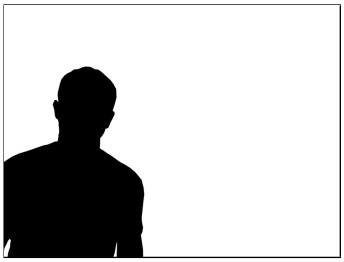
In Figure B.2 we show some sample images in the Challenge training dataset along with respective ground truth masks and attacks which have been performed upon.

(a) Sample image.



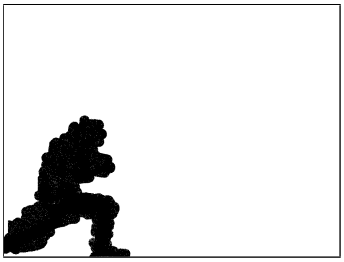
(b) Ground truth mask. Copy-move: the plant has been duplicated.

(c) Sample image.



(d) Ground truth mask. Splicing: the person belongs to another image.

(e) Sample image.



(f) Ground truth mask. Healing Brush: a crouching person has been removed.

Figure B.2: Sample fake images in $\mathcal{I}$.