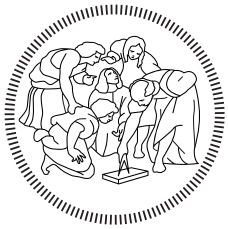


Politecnico di Milano
Dipartimento di Elettronica, Informazione e Bioingegneria
Dottorato di Ricerca in Ingegneria dell'Informazione

Data-driven and handcrafted features for forensics analysis and source attribution

Doctoral Dissertation of:
Luca Bondi
ID: 848746

Advisor:
Prof. Stefano Tubaro
Tutor:
Prof. Matteo Cesana
Supervisor of the Doctoral Programme:
Prof. Barbara Pernici

XXXI edition

Abstract

The communication power associated with visual content makes digital images a powerful and effective tool to deliver messages, spread ideas, and prove facts.

Smartphones, digital cameras, and camcorders are becoming more affordable every day, and thus constitute a rapid and convenient way of capturing and sharing photos quickly and inexpensively. The increasing diversity of brands, models, and devices makes the creation of new visual contents easier every day, while the ever-growing access to social network and picture sharing platforms poses a set of challenges, from the diffusion of illegal content to copyright infringement.

The wide availability and ease of use of image manipulation software makes the process of altering an image simple and fast. This could severely reduce the trustworthiness of digital images for users, legal courts, and police investigators. The fake news phenomenon is a well-known and widespread example of the malicious use of digital pictures and manipulation software. Modified images done with precision are used to create false proofs for made-up stories, exploiting the often unquestionable trust with which readers take in visual content.

In this thesis we face several challenges related to the analysis of digital images. A first step in assessing image authenticity, and tracing an image back to its origins, consists in determining which device shot a specific picture. State-of-the-art techniques based on Photo Response Non-Uniformity (PRNU) prove to be very effective in determining the specific sensor that shot a picture. However, given the highly increasing number of devices, a full-range search over all the existing devices is impractical and time consuming. One of the ways to reduce the search space is to first find the camera model that took a picture, then test the image under analysis against the devices from the same camera model. In this thesis we present the first data-driven method designed to learn camera model features directly from a collection of images, showing how modern deep-learning techniques based on Convolutional Neural Networks (CNN) can be adapted to multimedia forensics tasks.

When it comes to a large-scale search of picture-device matches based on PRNU, at least two challenges arise: time and storage space constraints. To address such challenges,

the forensics community explored a series of techniques to compress PRNU fingerprints and residuals. In order to reduce storage space requirements, while lowering the computational complexity, we introduce two techniques to address PRNU compression, by exploiting classical signal processing analysis and data reduction techniques.

While determining the origin of a digital image is important to solve copyright infringement cases, digital images can be locally altered by adding, removing, or modifying objects with the goal of changing the semantics of the image. We present how to exploit the features learned with a CNN trained for camera model identification with the goal of detecting and localizing tampered regions within an image.

Under both device identification and camera model identification perspectives, we study a set of possible antiforensics attacks tailored at anonymizing an image to prevent the correct identification of its origin. This allows us to understand the limitations and weaknesses of the proposed camera model and device identification techniques.

Finally, we leverage the knowledge and skills acquired in mixing together handcrafted signal processing and data-driven methods in two different forensics applications: Laser Printer Attribution and Single versus Double JPEG Detection. In both scenarios the key to tackle the forensics task at hand is fusing together a proper signal pre-processing technique with a carefully designed data-driven system.

Contents

1 Introduction

Since the beginning of human history, visual communication has had a primary role in the diffusion of ideas, concepts, religions, and laws. Rural pictures, stone engravings, paintings, drawings, sculptures, and monuments were the universal language across cultures to preserve and to hand down moments of history in a way that was accessible to everyone. In 1839 when photography was invented, the importance of visual content greatly increased. The fact that human-made machinery could impress a single moment in time on a film and preserve it possibly forever, made photography the trusted method of conserving memories and spreading indisputable facts. This combination of communicative power and intrinsic trust attributed to photography made images the ideal medium to be doctored with various intents, ranging from political reasons to marketing purposes. In most cases, the only way to determine if and how the image content was modified was to have access to different versions of the image itself, to spot missing, added, or modified persons, faces, and objects [1].

The digital revolution started in the 2000s made the diffusion and impact of images on society even more dramatic than it was in the past. Blog posts, tweets, and online magazine articles are all rapid and inexpensive ways of sharing material with a very broad audience. This wide and legit diffusion of digital contents has the counter-side effect of allowing easy access to digital copies of images and videos to malicious actors, willing to modify, redistribute and change the semantic of multimedia contents. Pictures protected by copyright can easily be used and distributed without giving credit to their owners. Well-prepared fake news can be enriched by un-authentic pictures to sway the public opinion in a more effective way. Moreover, in recent years creating forgeries on digital images no longer requires advanced technical knowledge, due to the extensive availability of image processing software suites. The widespread, uncontrolled, and fast diffusion of possibly modified images through social media, blogs, and websites calls for the development of accurate and possibly scalable image forensics tools.

Authenticity verification, computer-generated image detection, tampering detection and localization, and source device identification are just a few examples [2] of the open topics that researchers from the multimedia forensics community have addressed in the

last decade. The scientific community itself is not immune from image forgeries [3], thus methods to perform automated analyses of image contents are even more necessary today than in the past. Moreover, the recent introduction of Generative Adversarial Networks [4] paves the way to a completely new generation of image modifications, in between computer-generated graphic and handcrafted artifacts, as seen in the well-known case of Face-Swap [5].

The series of techniques developed in the information and multimedia forensics field [6–8] to assess the trustworthiness of digital content can be broadly divided into two categories: active and passive methods. The former requires that at the source side (i.e. digital camera, smartphone) a watermark is injected or a signature is computed at acquisition time. Any successive modification of the image will alter the signature or watermark, thus allowing a forensic investigator, who knows the original watermark/signature, to detect the tampering. The latter family of methods is instead solely based on the analysis of the content under investigation, without the need of any prior knowledge about the source. The main idea is to analyze the bitstream and the pixel content of the image and infer anomalies to extract significant "forensics features" whose behavior is affected by any intermediate manipulation occurred between the acquisition and the analysis.

Despite the effectiveness of active forensic methods, the majority of multimedia content spread over the Internet comes from unknown and uncontrolled sources. Reporters willing to protect their identities against possible prosecutors, malicious content creators, and criminals not willing to be found are certainly avoiding and possibly erasing any sort of watermark or signature eventually inserted at acquisition time by the sensing device. Following these premises, the number of passive techniques willing to exploit intrinsic signatures embedded in digital contents is constantly expanding, to accommodate the needs of courts, police officers and investigators willing to trace the origins of contents under investigation.

Within the set of passive methods, in this thesis we will focus specifically on the family of passive source identification methods. In particular, image source identification aims at answering a set of different questions: i) Which camera make/model shot this picture? ii) Did any of these devices shoot this picture? iii) Did this specific device shoot this picture?

When examining an image under analysis, a forensic investigator might have hints about the camera model or even the specific device that shot the picture and want to know what is the confidence of such a hypothesis. Image source attribution and verification are approached using data-driven and statistical frameworks, to address camera model identification and device sensor verification, respectively.

The forensics traces used to solve the problem of source attribution can also be used and adapted to solve other forensics tasks, such as tampering detection and localization. Local inconsistencies in forensics features extracted from different portions of an image can be exploited to determine which regions of an image have possibly been modified or come from a different camera model or from a different device.

Among the problems still open to the practical deployment of forensics systems, large-scale search over sensor databases [9] poses interesting challenges in the trade-off between performance and recognition accuracy. A large database system, that possibly has to store hundreds of thousands of sensor fingerprints, needs an efficient compression method tailored to the nature of the fingerprints. When it comes to match those fingerprints with query data, efficient matching algorithms are the key to provide fast and accurate answers to investigators and scientists. While compression and fast-matching are vital, the cost in terms of matching accuracy should be the smallest possible.

For both camera and device identification we also face the antiforensics perspective, developing methods to anonymize contents in such a way that a forensic analyst is no longer able to correctly detect the origin of an anonymized picture. This allows further development and refinement of the image source attribution technique at hand, by being aware of the possible set of anonymization attacks.

The data-driven techniques developed and studied for source attribution problems are then adapted in other forensics applications, such as laser printer attribution and recognition of image compression history.

The common question behind passive source identification systems is thus the following: What is the best feature set we can find, for some specific input signal, that preserves the forensics information of interest while reducing noise and content-related components?

With a set of data-driven techniques developed ad-hoc for the forensics tasks at hand, together with more classic handcrafted features and signal processing techniques, in this thesis we show how to face some of the open challenges in the multimedia forensics field.

1.1 Original contributions

The main contributions of this thesis are related to source identification of digital images and the application of data-driven approaches in multimedia forensics.

The first part of the thesis is devoted to the problem of camera model identification. In 2017 we proposed [10] the first method using a Convolutional Neural Network (CNN) to solve the problem of camera model identification, investigating some of the challenges

that data-driven methods should face when used for forensics applications. The features learned by such CNN are then used to detect forged images and localize the tampered regions [11]. An antiforensics approach attacking CNN for camera model identification was developed in [12], in order to test the limits and the drawbacks of deep-learning approaches to camera model identification.

The second part of the thesis tackles the problem of Photo Response Non Uniformity [13] (PRNU) compression. All the digital images acquired with CCD/CMOS matrix sensors, i.e. those sensors built in DSLR cameras, smartphones, etc., have an embedded noise fingerprint due to manufacturing imperfections in sensor's silicon. This characteristic fingerprint can be properly extracted and used to identify the specific sensor that acquired a picture. By correlating the extracted noise residual from an image under investigation with the reference fingerprint extracted from a set of flatfield images acquired with the same sensor, a forensic investigator can easily determine if the the picture comes from that device or not. When it comes to store or send over a band-limited communication channel either the fingerprint or the residual, the choice of the proper lossy compression system becomes the key to scale PRNU-based device matching to large scale scenarios. Two antiforensics attacks are proposed, one based on an inpainting techniques and the other based on a properly designed double-input autoencoder.

The last part includes two contributions in the multimedia forensics field. The expertise acquired in the use of Convolutional Neural Networks for forensics purposes, together with more classical signal processing techniques derived from device identification, are merged to propose innovative solutions. An improvement to existing single versus double JPEG compression detectors is proposed by feeding a CNN with handcrafted features extracted from the raw image. Laser printer attribution is faced with a combination of CNNs whose inputs are differently pre-procesed versions of the same letters.

In the following, we detail the original contributions for each of the topics.

Camera model identification

The first step in order to determine the origin of an image, is assessing the make and model of the device that shot the image. This allows to refine the search on a set of specific devices and to exclude a large set of improbable candidates. Traditionally, camera model identification was faced by extracting a set of handcrafted features from the image then fitting a statistical model or a machine learning classifier on top of the features. One of the main drawbacks of such approaches is the fact that the recognitions performance quickly drop for image patches smaller than 256×256 pixels. In [10] we present the first Convolutional Neural Network architecture that proves to work on image

patches as small as 64×64 pixels. Given the reduced dimensions, the choice of patches whose content is textured enough and not saturated proves to be the key to correctly solve camera attribution. In an extended study presented in [14], we evaluated the need of proper training, validation, and test protocols when dealing with CNN for camera model identification. In particular, we show that with such small image patches the risk of the CNN learning the content of the pictures instead of meaningful forensics traces is negligible. We also propose a data splitting policy that avoids the CNN to learn device specific features, by training and testing on different devices from the same camera models. Moreover, with such a small patch size, we show that there is no need to resort to deep architectures, as the recognition performance do not improve with respect to shallow CNN architectures.

Tampering detection and localization

One of the main motivations behind the choice of 64×64 pixels patches for camera model identification, lies in the fact that being able to correctly recognize the camera model in small portions of an image opens the way to tampering localization based on camera model inconsistencies. In [11] we present an iterative clustering method for tampering detection and localization based on the camera model features extracted thanks to the CNN presented in [10]. One of the key points of the proposed approach is the use of a patch quality measure in order to assess "how much a single patch can be trusted". In facts, features extracted from dark or saturated patches cannot be trusted as much as features extracted from textured regions. This observation, together with the definition of a proper metric to clusterize the feature vectors from the patches, allows to tackle in a successful way the tampering detection and localization problem when an image is spliced with a content coming from a different camera model.

Camera model antiforensics

Due to their linear nature, CNNs classifiers are vulnerable to antiforensics attacks, as shown in [4]. Un-noticeable artifacts can dramatically change the forensics properties of an image, so that a CNN-based classifier fails in correctly detecting the original camera model. In [12] we presented a framework to attack a CNN-based camera model classifier. We show how even deep architectures are affected by this vulnerability. The two used methods, Fast Gradient Sign Method (FGSM) and Jacobian-based Saliency Map Attack (JSMA), are able to craft adversarial images without requiring access to the CNN used to classify the image.

PRNU compression

Dealing with a large number device fingerprints for device source identification poses several practical problems in terms of storage and computational requirements. Binarized gaussian random projections [15] proved to be a very effective method to reduce the dimensionality of PRNU fingerprints and residuals. In [16] we propose an improved processing chain composed by decimation, gaussian random projections, ternary quantization and coding tailored to increase the compression rate while preserving the highest possible matching accuracy. Exploiting the artifacts generated by JPEG compression on images, we are able to greatly reduce the required bitrate necessary to store or send a fingerprint. While the approach presented in [16] focuses on the improvement of the compression rate, in [17] we proved how a careful choice of a sub-sampling pattern together with an embedding policy, allows to avoid the use of random projections while preserving the same compression rate and performance. This allows a reduction to less than 0.1% in terms of computational complexity.

PRNU applications

One of the applications where PRNU fingerprints are exploited is device authentication. In [18] we proposed a method to extend the concept of device fingerprint, by embedding traces from the camera, i.e. PRNU, the accelerometer, the gyroscope and the magnetometer to obtain a forensic fingerprint for a smartphone.

PRNU traces can also be exploited in video phylogeny applications [19] to refine the search for a cluster of video coming from the same source. In facts, a system based solely on content information will group together videos whose semantic is similar, while for phylogeny trees reconstruction it is of utmost importance to group together only those videos that share a common source device.

PRNU antiforensics

As for camera model identification, also when it comes to device identification a set of existing techniques to anonymize pictures by removing PRNU traces are well known from the literature [20]. In [21] we propose an inpainting-based solution to camera anonymization. Every image is sub-sampled to create a set of sub-images. For each sub-image the missing pixels are reconstructed solving an inverse optimization problem. Finally the sub-images are merged together to rebuild an image whose pixel values have all been interpolated. This technique allows to preserve a high image quality while almost completely removing the traces of PRNU. Another approach in terms of PRNU

antiforensics is presented in [22]. This represents the first PRNU anonymization approach carried out with a Convolutional Auto-Encoder. An image and the reference camera fingerprint are used together in an anonymization loop to create a new version of the image visually indistinguishable from the original one. Even though an antiforensics-aware investigator could spot the attack, this technique constitutes a first important step toward the usage of data-driven methods also in device anonymization.

Data-driven approaches in multimedia forensics

The expertise and the techniques developed in camera and device identification gives us the tools to cope with other multimedia forensics applications exploiting several data-driven methods.

The challenging problem of detecting whether a JPEG image has been compressed one or multiple times is still an open question when it comes to lightly compressed images. In [23] we propose a single versus double JPEG detection pipeline based on the idea that pre-processing the image before feeding it to a CNN for classification greatly improves the recognition performance. In particular, exploiting PRNU residuals and DCT coefficient histograms as feature maps in input to several CNNs we obtain state-of-the-art performance when dealing with lightweight JPEG compressed images.

Another forensic application where data-driven method prove to perform at par or better than the state of the art is laser printer attribution. In [24] we present a CNN-based system for printer recognition. Documents printed from ten different printers are classified based on a double fusion scheme. Letters “a”,“e” and “o” are extracted from each document, pre-processed, then fed to different CNN trained to extract meaningful features to the task at hand. The feature vectors extracted for each different pre-processing policy are concatenated and used together as a single feature for a multi-class ensemble classifier. Finally, majority voting is used to aggregate at document level the predictions obtained from each letter.

In the two aforementioned forensics applications the key part of the work is always mixing the classical signal processing and statistical knowledge with the data-driven tools provided by the machine learning community. Thanks to this hybrid approach we show that forensics tasks, traditionally addressed with handcrafted features and statistical models, can benefit of the availability of large collections of labeled data to improve recognition performance while preserving robustness.

1.2 Outline

In order to ease the reading and present the topics in an organic way, we organize this thesis in three main chapters.

Chapter 2 presents the methodologies and the results from [10–12, 14]. In Section 2.4 we present the CNN-based method proposed in [10, 14] to tackle the camera model identification problem, discussing the types of architectures, the choice of training data and the splitting policies to train the CNN in an unbiased way. Section 2.5 introduces the method presented in [11] to address the detection of tampered images and the localization of forgeries through an iterative clustering approach based on the features extracted with the CNN trained for camera model identification. The counter-forensics attack against CNN-based camera model identification proposed in [12] is presented in Section 2.6.

Chapter 3 introduces the proposed PRNU compression techniques, the antiforensics attacks and two PRNU-related applications. Section 3.4 describes the projection design methodology and the compression pipeline introduced in [16,17], explaining the rationale, the models and evaluating the choice of several parameters. Two PRNU-anonymization attacks from [21, 22] are introduced in Section 3.5. Two applications of PRNU-based device identification [18, 19] are presented in Section 3.7.

Chapter 4 gives to the reader an overview of two forensics applications where a combination of classical signal processing techniques, handcrafted features and data-driven methods are fused together to tackle the tasks at hand. Section 4.1 presents the work published in [23] about single versus douple JPEG compression detection with a combination of CNNs and handcrafted pre-processing operations. Section 4.2 reports the method proposed for laser printer attribution based on single characters classification. Exploiting early fusion of differently pre-processed CNN features, and late fusion of scores from different letters within a single document, we present a pipeline able to correctly classify printed documents generated by ten different laser printers.

Final considerations and conclusions are drawn in Chapter 5.

1.3 Notation

For the sake of clarity, in the following vectors are given by boldface letters, e.g., $\boldsymbol{x}$, and are considered to be column vectors. The ith sample of $\boldsymbol{x}$ is represented by x_i. The Hadamard (sample-wise) product between $\boldsymbol{x}$ and $\boldsymbol{y}$ is denoted by $\boldsymbol{x} \circ \boldsymbol{y}$. Matrices are denoted by bold capital letters, e.g., $\boldsymbol{X}$, and the i, jth element of $\boldsymbol{X}$ is indicated by a subindex, e.g., $X_{i,j}$. The Hadamard product between $\boldsymbol{X}$ and $\boldsymbol{Y}$ is denoted by $\boldsymbol{X} \circ \boldsymbol{Y}$. The

Table 1.1: Table of symbols

Symbol	Description
$\boldsymbol{x}$	column vector
x_i	ith sample of $\boldsymbol{x}$
$\boldsymbol{X}$	matrix
$X_{i,j}$	i,jth element of $\boldsymbol{X}$
$\boldsymbol{X} \circ \boldsymbol{Y}$	Hadamard product between $\boldsymbol{X}$ and $\boldsymbol{Y}$
$\boldsymbol{X} \otimes \boldsymbol{Y}$	Kronecker product between $\boldsymbol{X}$ and $\boldsymbol{Y}$
$\boldsymbol{X}\boldsymbol{Y}$	matrix multiplication between $\boldsymbol{X}$ and $\boldsymbol{Y}$
$\boldsymbol{I}$	identity matrix
$\mathbf{0}$	all-zeros matrix
$\mathbf{1}$	all-ones vector

Kronecker product between two matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$ is denoted by $\boldsymbol{X} \otimes \boldsymbol{Y}$. The matrix multiplication between $\boldsymbol{X}$ and $\boldsymbol{Y}$ is denoted by $\boldsymbol{X}\boldsymbol{Y}$. Given a matrix $\boldsymbol{X}$, its column-wise unwrapped vector is denoted by $\boldsymbol{x}$. $\boldsymbol{I}$, $\mathbf{0}$ and $\mathbf{1}$ denote the identity matrix, the all-zeros matrix, and the all-ones vector, respectively. Table 1.1 summarizes the symbols used afterwards.

1.4 List of included publications

The work presented in this thesis includes original contents from the following peer-reviewed papers, published by the author in international journals and conferences over the past four years.

- [10] L. Bondi, L. Baroffio, D. Güera, P. Bestagini, E. J. Delp and S. Tubaro, *"First Steps Toward Camera Model Identification With Convolutional Neural Networks"*, in IEEE Signal Processing Letters, vol. 24, no. 3, pp. 259-263, March 2017.

- [14] L. Bondi, D. Güera, L. Baroffio, P. Bestagini, E. J. Delp and S. Tubaro, *"A preliminary study on convolutional neural networks for camera model identification"*, Electronic Imaging, Media Watermarking, Security, and Forensics 2017, pp. 67-76

- [11] L. Bondi, S. Lameri, D. Güera, P. Bestagini, E. J. Delp and S. Tubaro, *"Tampering Detection and Localization Through Clustering of Camera-Based CNN Features"*, 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Honolulu, HI, 2017, pp. 1855-1864.

- [12] D. Güera, Y. Wang, L. Bondi, P. Bestagini, S. Tubaro and E. J. Delp, *"A Counter-Forensic Method for CNN-Based Camera Model Identification"*, 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Honolulu, HI, 2017, pp. 1840-1847.
- [17] L. Bondi, F. Pérez-González, P. Bestagini and S. Tubaro, *"Design of projection matrices for PRNU compression"*, 2017 IEEE Workshop on Information Forensics and Security (WIFS), Rennes, 2017, pp. 1-6.
- [16] L. Bondi, P. Bestagini, F. Pérez-González and S. Tubaro, *"Improving PRNU Compression through Preprocessing, Quantization and Coding"*, in IEEE Transactions on Information Forensics and Security, vol. 14, no. 3, pp. 608-620, March 2019
- [21] S. Mandelli, L. Bondi, S. Lameri, V. Lipari, P. Bestagini and S. Tubaro, *"Inpainting-Based camera anonymization"*, 2017 IEEE International Conference on Image Processing (ICIP), Beijing, 2017, pp. 1522-1526.
- [22] N. Bonettini, L. Bondi, D. Güera, S. Mandelli, P.Bestagini, S. Tubaro and E.J. Delp *"Fooling PRNU-Based Detectors Through Convolutional Neural Networks"*, 2018 26th European Signal Processing Conference (EUSIPCO), Rome, 2018, pp. 1-6
- [18] I. Amerini, P. Bestagini, L.Bondi, R. Caldelli, M. Casini and S. Tubaro *"Robust smartphone fingerprint by mixing device sensors features for mobile strong authentication"*, Electronic Imaging, Media Watermarking, Security, and Forensics 2016, pp. 1-8
- [19] S. Lameri, L. Bondi, P. Bestagini and S. Tubaro, *"Near-duplicate video detection exploiting noise residual traces"*, 2017 IEEE International Conference on Image Processing (ICIP), Beijing, 2017, pp. 1855-1864
- [23] M. Barni, L. Bondi, N. Bonettini, P. Bestagini, A. Costanzo, M. Maggini, B. Tondi and S. Tubaro, *"Aligned and non-aligned double JPEG detection using convolutional neural networks"*, Journal of Visual Communication and Image Representation, vol. 49, pp. 153-163, 2017.
- [24] A. Ferreira, L. Bondi, L. Baroffio, P. Bestagini, J. Huang, J. A. dos Santos, S. Tubaro and A. Rocha, *"Data-Driven Feature Characterization Techniques for Laser Printer Attribution"*, in IEEE Transactions on Information Forensics and Security, vol. 12, no. 8, pp. 1860-1873, Aug. 2017.

2 Methods based on camera model traces

2.1 Introduction

The rapid proliferation of inexpensive image capturing devices has determined a widespread diffusion of digital pictures on the web. Due to the increasing availability of image acquisition devices and multimedia sharing platforms, photos are becoming a pervasive part of our daily life. As downloading, copying, forging, and redistributing digital material is becoming easier over the years, some forms of regulation and authenticity verification have become of urgent necessity. Moreover, the increase in the number of digital images that are being uploaded and shared online has given rise to unique privacy and forensic challenges [25].

As sharing any kind of photographs through websites and social media is at everyone's hand, verifying the veracity and authenticity of diffused images is far from being an easy task [1, 26]. To this purpose, the forensic community has developed a wide series of tools to reverse-engineer the history of multimedia objects and assess the trustworthiness of digital images [6, 7].

Camera model identification

Among the problems tackled by the image forensics community, one that is still under the spotlight due to its implications in several applications is camera model identification [1, 27–29]. This is, given an image, detect which camera model and brand has been used to shoot it. Indeed, detecting the camera model used to shot a picture can be crucial for criminal investigations and trials. This information can be fruitfully exploited for solving copyright infringement cases, as well as for pointing out the authors of distributed illicit material (e.g., pedo-pornographic shots). Even when deeper source identification granularity is needed (i.e., detecting the unique camera instance rather than just the make and model), camera model identification can be considered an important preliminary step to narrow down the set of investigated camera instances [29]. Moreover, being

able the detect the used camera model by analyzing small image patches is a possible way to expose splicing operations [30] operated between images coming from different cameras [31, 32].

Attribution of a picture to a specific camera model in a blind fashion (i.e., not leveraging watermarks introduced at photo inception) is possible exploiting intrinsic artifacts left at shooting time by the acquisition process. The rationale behind blind state-of-the-art camera model identification detectors is that each camera model performs peculiar operations on each image at acquisition time (e.g., different JPEG compression schemes, proprietary algorithms for CFA demosaicing, etc.). The idea is that each camera model implements a series of characteristic complex operations at acquisition time, from focusing light rays through lenses, to interpolation of color channels by proprietary demosaicing filters, to brightness adjustment, and also others. As these operations are non-invertible, they introduce some unique artifacts on the final image. These artifacts are footprints that act as an asset to detect the used camera model.

To this purpose, several camera model identification algorithms have been proposed in the literature. An overview of the state-of-the-art techniques is presented in Section 2.3.

A common aspect of all the existing algorithms is that they rely on manually defined procedures to expose traces characterizing different camera models. This means that they rely on some model assumptions made "a priori". However, recent advancements established by deep learning techniques in computer vision [33, 34] have shown that it is possible to improve the accuracy in detection and classification tasks by taking advantage of great amounts of data in order to learn characteristic features directly from the data itself. However, only starting from 2012 [35] the availability of fast and parallel computing devices (in particular GPUs) really made CNNs at researchers' hands. AlexNet [35], Network in Network [36] and GoogLeNet [37] are just three examples of data-driven Deep Learning models that showed impressive accuracy improvements in image classification and localization tasks over the previously widespread handcrafted features [38–40]. The use of deep learning techniques for image and video classification tasks [41–43] has shown that it is also possible to learn characteristic features that model a problem space directly from the data itself. These methods are known as data-driven, as they learn directly from data rather than following an analytic model.

Considering that this feature learning paradigm has recently proved fruitful also for forensics applications [44–48], in Section 2.4 we investigate the use of feature learning techniques in the camera model identification context. Our objective is to show that it is possible to use Convolutional Neural Networks (CNNs) to learn discriminant features directly from the observed known images, rather than relying on hand-crafted descriptors.

In principle, this enables to possibly capture also characteristic traces left by non-linear and difficult to model operations during the acquisition pipeline. In particular, we make use of a CNN to capture camera-specific artifacts in an automated fashion, and Support Vector Machines (SVMs) for classification. We investigate the behavior of different CNN architectures to select a proper network for discriminant feature learning on $64 \times 64 \times 3$ (i.e., height $\times$ width $\times$ colors) image patches, while keeping computational complexity at bay. In particular, we compare a series of CNN architectures differing in the number of convolutional, pooling, fully-connected and rectified linear unit (ReLU) layers. For each type of architecture, several hyper-parameters choices (kernel size, stride, number of feature maps, etc.) are examined. Moreover, we focus on the importance of a proper training protocol, which is essential to make sure that the CNN learns important characteristics (e.g., properties discriminating camera models) rather than biased information (e.g., the semantic of the captured scenes). To this purpose, strongly inspired by [29], we consider different amounts of training images, depicting either the same or different scenes, and show how different training choices affect classification results.

Tampering detection and localization

Among the many techniques developed in the image forensic literature, some are tailored to the detection of a single trace left by a specific operation on the whole picture (i.e. tampering detection). Other techniques focus on localizing regions that present traces of editing (i.e. tampering localization). In Section 2.3.2 we provide a review of state-of-the-art techniques in tampering detection and localization. In Section 2.5 we address tampering detection and localization problem for images obtained through splicing of content originally shot from different camera models. Specifically, we analyze images in a patch-wise fashion, and aim to detect whether different patches belong to different camera models (i.e. the image has been forged) exploiting descriptors learned by the CNN presented in Section 2.4. These descriptors extracted from small image patches are analyzed by means of an iterative clustering technique to expose regions of an image that appear to be obtained from different camera models.

Camera model antiforensics

With the introduction of CNNs as detectors for camera model identification, a new vector for counter-forensic attacks is presented for a malevolent skilled individual. The idea of counter-forensics was first introduced in [49], where the authors presented the concept of fighting against image forensics with a practical application, namely a method

for resampling an image without introducing pixel correlations. Before exploring the vulnerabilities of CNN-based camera model detectors, it is important to note that detectors that rely on hand-crafted features are not immune to similar counter-forensics attacks. As explained in [50], digital camera fingerprints are vulnerable to forging. In particular, if an attacker obtains access to images from a given camera, they can estimate its fingerprint and "paste" it into an arbitrary image to make it look as if the image came from the camera whose fingerprint was stolen. An early attempt to investigate such counter-forensic methods appeared in [51]. As presented in [52], several machine learning models, including state-of-the-art CNNs, are vulnerable to adversarial attacks. This means that these machine learning models mis-classify images that are only slightly different from correctly classified images. In many cases, an ample collection of models with different architectures trained on different subsets of the training data mis-classify the same adversarial example [53]. Although there are techniques such as adversarial training [52] or defensive distillation [54] that can slightly reduce the incidence of adversarial examples in CNN-based detectors, defending against adversarial examples is still an on-going challenge in the deep learning community. Adversarial attacks are hard to defend against because they require machine learning models that produce correct outputs for every possible input. The imposition of linear behavior when presented with inputs similar to the training data, though desirable, is precisely the main weakness of CNNs [53]. Due to the massive amount of possible inputs that a CNN can be presented with, it is remarkably simple to find adversarial examples that look unmodified to us but are misclassified by the network. Designing a truly adaptive defense against adversarial images remains an open problem.

In Section 2.6, we propose a counter-forensic method to subtly change an image to induce an error in its estimated camera model when analyzed by a CNN-based camera model detector. We test our counter-forensic method, using two well established adversarial image crafting techniques [53, 55], against an advanced deep learning architecture [56] carefully trained on a reference camera model dataset. Our results show that even modern and properly trained CNNs are susceptible to simple adversarial attacks.

2.2 Background

In this section, we provide an overview of the digital images acquisition process, together with the background on Convolutional Neural Networks (CNNs) and CNN counter-forensics methods sufficient to understand the rest of the chapter.

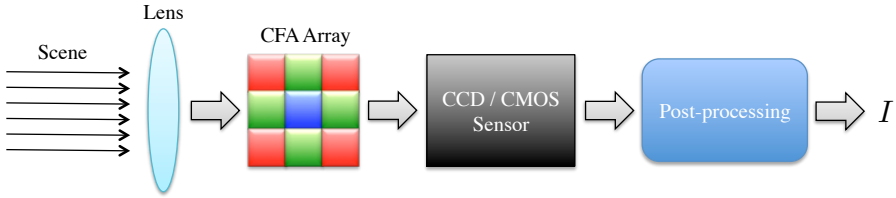


Figure 2.1: Digital image acquisition pipeline. Ray-lights reflected from the scene are focused by the lens on a Color Filter Array superimposed to a CCD/CMOS sensor. The sensor output is processed and an RBG image is produced as output.

2.2.1 Digital image acquisition

Digital camera acquisition pipeline is a complex chain of operations typically defined by the sketch in Figure 2.1. According to this diagram, light rays reflected by the scene are collected by a lens that focuses the rays on a CCD/CMOS sensor. The nature of the lens, the focal length, the aperture of the diaphragm modify the light-field that reaches the sensor, leaving peculiar traces [57]. The sensor itself is a matrix of pixels, whose sizes are intrinsically not perfectly equal. This makes every pixel have a slightly different gain, thus embedding in the signal a multiplicative weak noise [13]. On top of the sensor a Color Filter Array (CFA) is typically used to filter the light that hits every pixel, so that some of the sensor pixels are just hit by red, green or blue light components. At this stage, each pixel of the RAW image contains just the color information related to a specific portion of the electromagnetic spectrum. To generate a full-size RGB image every pixel is interpolated in the red, green, blue components, with an operation call "demosaicing". This operation leaves traces that are peculiar of the manufacturer-proprietary algorithm and thus could be used for forensics purposes [27, 58]. Additional operations as white-balancing, gamma correction and exposure adjustment are part of the camera's firmware and contribute at generating the final 8-bit RGB image usually encountered in digital imaging. A last important step in the post-processing chain is JPEG compression, to reduce the storage space necessary to store the pictures. As JPEG implementations are custom-made by camera manufacturers, the traces left by different compression algorithms can be exploited in forensics applications [59].

2.2.2 Convolutional Neural Networks

Deep learning and in particular CNNs recorded amazingly good performance in several computer vision applications like image classification, face recognition, pedestrian detection and handwriting recognition [33]. A CNN is a complex computational model partially inspired by the human neural system that consists of a very high number of

interconnected nodes, or neurons. Connections between nodes have a numeric weight parameter that can be tuned based on experience, so that the model is able to learn complex functions. The nodes of the network are organized in multiple stacked layers, each performing a simple operation on the input. The set of operations in a CNN typically comprises convolution, intensity normalization, non-linear activation and thresholding, and local pooling. By minimizing a cost function at the output of the last layer, the weights of the network (e.g., the values of the layers' filters) are tuned so that they are able to capture patterns in the input data and automatically extract distinctive features.

Differently from traditional "handcrafted" feature extraction algorithms, in which the feature extraction process is driven by human intuitions, in CNNs the feature extraction process is driven by data. Such complex models are trained resorting to backpropagation, coupled with an optimization method like gradient descent, and large annotated training datasets. The first layers of the networks usually learn low-level visual concepts like edges, simple shapes and color contrast, whereas deeper layers combine such simple information to identify complex visual patterns. Finally, the last layer consists of a set of data that are combined to define a given cost function that needs to be minimized. For instance, in the context of image classification, the last layer is composed of N nodes, where N is the number of classes, that define a probability distribution over the N visual category. That is, the value of a given node $p_i,\ i = 1, \ldots, N$ belonging to the last layer represents the probability of the input image to belong to the visual class c_i. Depending on user choices, it is possible to select the class maximizing p_i as classification result, or to use all p_i values as feature vector to train an external classification tool (e.g., a Support Vector Machine).

To train a CNN model for a specific image classification task we need:

1. to define the metaparameters of the CNN, i.e., the sequence of operations to be performed, the number of layers, the number and shape of the filters in convolutional layers, etc;

2. to define a proper cost function being minimized during the training process;

3. to prepare a (possibly large) dataset of training and test images, annotated with labels according to the specific tasks.

Figure 2.2 reports a minimalistic example of a small CNN architecture depicting some of the commonly used layers. To better understand the role of each one of them, in the following we report a description of the most common building blocks, called layers. Each layer $\mathcal{L}_i$ takes as input either an $H_i \times W_i \times P_i$ feature map or a feature vector of

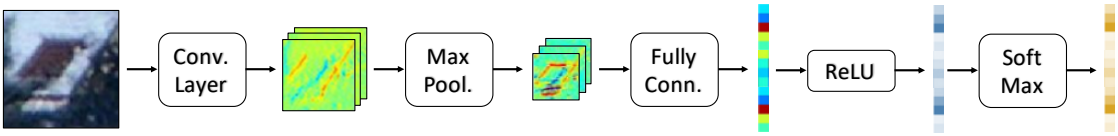


Figure 2.2: Simple CNN architecture consisting of commonly used layers. A small color image patch is processed through convolutional, max pooling, fully-connected and ReLU layers. Finally, SoftMax is applied to obtain a feature vector.

size P_i and produces as output either an $H_{i+1} \times W_{i+1} \times P_{i+1}$ feature map or a feature vector of size P_{i+1}. Layer types used in the following are:

- **Convolutional layer** Performs convolution, with stride S_h and S_w along first two axes, between input feature maps and a set of P_{i+1} filters with size $K_h \times K_w \times P_i$. Output feature maps have size $H_{i+1} = \lfloor \frac{H_i - K_h + 1}{S_h} \rfloor$, $W_{i+1} = \lfloor \frac{W_i - K_w + 1}{S_w} \rfloor$ and P_{i+1}.

- **Max-pooling layer** Performs maximum element extraction, with stride S_h and S_w along first two axes, from a neighborhood of size $K_h \times K_w$ over each 2D slice of input feature map. Output feature maps have size $H_{i+1} = \lceil \frac{H_i - K_h + 1}{S_h} \rceil$, $W_{i+1} = \lceil \frac{W_i - K_w + 1}{S_w} \rceil$ and $P_{i+1} = P_i$.

- **Fully-connected layer** Performs dot multiplication between input feature vector (or flattened feature maps) and a weights matrix with P_{i+1} rows and P_i (or $H_i \cdot W_i \cdot P_i$) columns. Output feature vector has P_{i+1} elements.

- **ReLU layer** Performs element-wise non linear activation. Given a single neuron x, it is transformed in a single neuron y with $y = \max(0, x)$.

- **Softmax layer** Turns an input feature vector to a vector with the same number of elements summing to 1. Given an input vector x with P_i neurons x_j $i \in [1, P_i]$, each input neuron produces a corresponding output neuron $y_j = \frac{e^{x_j}}{\sum_{k=1}^{k=P_i} e^{x_k}}$.

In a CNN, the weights of the convolutional and the fully-connected layers are learned using backpropagation [34] coupled with an optimization method such as gradient descent [60] and the use of large annotated training datasets. A loss functions is defined between the predicted and the expected output, giving rise to a prediction error and gradients to update the weights. The set of training samples is usually split into mini-batches [61]. For each mini-batch the predictions for each sample are first computed (forward phase), then the gradients with respect to the loss function are averaged to update the filters' weights (backward phase). This forward/backward process on a mini-batch is called "iteration". The use of mini-batches allows to fit the computation into the available memory, while

averaging over training samples to approximate an ideal gradient descent method that would require the knowledge of all the gradients from all the samples in the dataset.

Training data are usually split into a training set and a validation set. The former is used to drive the gradient descent method to update the filters' weights. The latter is used to compute the loss over a set of samples unknown to the training process, so to estimate the behavior of unseen data. Typically, the update of the filters is repeated multiple times across the training dataset. Each pass over a dataset is called "epoch". The training process is stopped when the loss computed on the validation set reaches its minimum. As discussed later, the choice of a proper train/validation/test split protocol can be very relevant for scientific purposes.

2.2.3 Deep learning in multimedia forensics

CNNs have been successfully used in recent years for many image recognition and classification tasks [33] and there are also many works which use CNN for applications of staganalysis, e.g. [48,62–64]. However, only recently, some works have started to explore CNNs for multimedia forensic applications.

One of the first works using CNNs for multimedia forensics is [45]. In this paper, the authors developed a detector for median-filtered images, whose capability of working on small 64×64 patches enabled its use also for tampering localization. In developing this algorithm, authors showed the importance of applying a pre-processing filtering step to images, in order to better expose forgery traces in a residual domain. The used CNN architecture is composed by only four convolutional layers and fewer inner-products, nonetheless providing very accurate results.

Based on the aforementioned works, it is worth noting how CNNs for multimedia forensics typically requires just a few layers to achieve promising results, being shallower than CNNs used for artificial intelligent and computer vision applications. This is probably due to the fact that the considered forensic classification tasks can be efficiently solved also using simple statistical model. This means that they do not need a very high abstraction capability, which proves essential for complex tasks like object recognition, in which the underlying model (i.e., human brain behavior) is particularly hard to describe.

2.2.4 Deep learning antiforensics

In [52] Szegedy et al. discovered that deep neural networks are prone to mis-classify with high confidence images contaminated with hardly perceptible perturbations. As an example, let us pick an image of a flower that is correctly classified by several neural

networks trained for object recognition. All the trained models, each one representing a different architecture, correctly classify the flower with very high confidence. If a specific low-power noise-like signal is added to the picture, all the trained models mis-classify the flower into a different category.

Two families of attacks are tailored to deep learning antiforensics. Targeted attacks aim at fooling a deep-learning model by having the model predict a specific target label for the adversarial image. Untargeted attacks are instead tailored to have the trained network mis-classify the input, regardless of the output label.

In the following, we present an untargeted technique presented by Goodfellow et al. in [53] known as Fast Gradient Sign Method(FGSM), and a targeted attack from Papernote et al. [55], namely Jacobian-Based Saliency Map Attack(JSMA).

Fast Gradient Sign Method

In [53], the fast gradient sign method was introduced for generating adversarial examples using the derivative of the loss function of the CNN with respect to the input feature vector. Given an input feature vector (e.g. an image), FGSM perturbs each feature in the direction of the gradient by magnitude ϵ, where ϵ is a parameter that determines the perturbation size. For a network with loss $J(\Theta, x, y)$, where Θ represents the CNN predictions for an input $\boldsymbol{x}$ and y is the correct label of $\boldsymbol{x}$, the adversarial example is generated as

$$\boldsymbol{x}^* = \boldsymbol{x} + \epsilon \text{sign}(\boldsymbol{\nabla}_x J(\Theta, \boldsymbol{x}, y)) \tag{2.1}$$

With small ϵ, it is possible to generate adversarial images that are consistently misclassified by CNNs trained on popular image classification datasets with a high success rate [53].

Jacobian-Based Saliency Map Attack

In [55], an iterative method for targeted misclassification was proposed. By exploiting the forward derivative of a CNN, it is possible to find an adversarial perturbation that will force the network to misclassify into a specific target class. For an input $\boldsymbol{x}$ and a convolutional neural network C, the output for class j is denoted $C_j(\boldsymbol{x})$. To achieve an output of target class t, $C_t(\boldsymbol{x})$ must be increased while the probabilities $C_j(\boldsymbol{x})$ of all other classes $j \neq t$ decrease, until $t = \arg\max_j C_j(\boldsymbol{x})$. This is accomplished by exploiting the

adversarial saliency map, which is defined as

$$s(\boldsymbol{x},t)_i = \begin{cases} 0 & \text{if } \frac{\partial C_t(\boldsymbol{x})}{\partial x_i} < 0 \text{ or } \sum_{j \neq t} \frac{\partial C_j(\boldsymbol{x})}{\partial x_i} > 0 \\ (\frac{\partial C_t(\boldsymbol{x})}{\partial x_i}) | \sum_{j \neq t} \frac{\partial C_j(\boldsymbol{x})}{\partial x_i} | & \text{otherwise} \end{cases} \tag{2.2}$$

for an input feature i. As we work with images, in our case each input feature i corresponds to a pixel i in the image input $\boldsymbol{x}$. Starting with a normal sample $\boldsymbol{x}$, we locate the pair of pixels $\{i, j\}$ that maximize $s(\boldsymbol{x},t)_i + s(x,t)_j$, and perturb each pixel by a constant offset ϵ. This process is repeated iteratively until the target misclassification is achieved. This method can effectively produce dataset examples that are correctly classified by human subjects but misclassified into a specific target class by a CNN with a high confidence.

2.3 State of the art

In this section we introduce the state of the art related to the topics presented in this chapter. In Section 2.3.1 we provide an overview of camera model identification systems, while Section 2.3.2 is devoted to tampering detection and localization methods. Finally Section 2.3.3 presents antiforensics techniques designed to attack camera model identification systems.

2.3.1 Camera model identification

In the last 15 years a number of systems were proposed to solve the problem of camera model identification. In the following we provide an overview of the most relevant ones. Some of the methods heavily rely on some prior model assumptions on the image acquisition pipeline, while other extract more general statistics as features set. Most of the methods proposed in the literature exploit some kind of machine-learning algorithm to classify or identify the camera models at hand.

In 2004 Kharrazi et al. [27] proposed a set of features to characterize camera models by extracting first, second and possibly higher order statistics from an image. The base assumption is that the Color Filter Array (CFA) configuration and color processing and transformations are unique to each camera manufacturer and model. The set of proposed features includes: i) the average pixel value for the color planes; ii) a measure of the correlation between the color planes; iii) a measure of the correlation between nearby pixels on each color plane; iv) the ratio of energy between R,G,B channels; v) the mean of Wavelet decomposition detail bands; vi) Image Quality Metrics as from [65]. The set

of 34 features extracted from each image are than used together with a Support Vector Machine (SVM) to classify the images coming from 5 different camera models.

Expanding the work in [27], in [58] Bayram et al. proposed to use interpolation traces as a feature for camera model identification. By exploiting the probability map and the weighting coefficients introduced in [66], Bayram et al. consider separately smooth and non-smooth areas of the image and apply a sequential forward floating search schema to the probability map magnitude spectrum, in order to select the best features from the given set of spatial frequency components. The selected features are then fed to an SVM classifier trained for camera model identification.

A completely different approach was followed by Choi et al. in [57]. The observation that radial lens distortions could serve as fingerprints for every camera model led the authors to propose a lens radial distortion estimation scheme based on Devernay's straight line method. The method exploits the fundamental property that the projection of every straight line in space onto a pinhole camera is a straight line. The parameters obtained from from aberration measurements are then used to train and test a SVM classifier.

In 2008 Filler et al. [28] studied the possibility of extracting meaningful features for camera model identification starting from the PRNU residual extracted from the image. The features are designed to reflect differences in the CFA demosaicing algorithm, and sensor signal transfer. 9 features are collected from the first 3 centralized sample statistical moments of the residual, for each color channel. The normalized cross-correlation between color channels is computed to get other 4 features after Principal Components Analysis (PCA). Residual block-wisez covariance and cyclic normalized autocorrelation of systematic errors of the in-camera post-processing algorithm in each row give rise to other 12 features. The collected features are then concatenated and used in a classification scheme powered by a SVM.

Cao et al. [67] proposed a framework to reversely classify the demosaiced samples into several categories. This allows to estimate the demosaicing formula for each category based on correlation models which can detect both intra and cross-channel demosaicing correlations. An Expectation-Maximization (EM) approach is used to solve the ambiguities of the demosaicing axes. A set of features is then computed based on the 16 demosaicing categories and the demosaicing reconstruction errors. Sequential features search is applied before feeding the best features to a Probabilistic-SVM classifier.

Xu et al. [68] proposed to use features derived from uniform gray-scale invariant local binary patterns (LBP). Considering 8-neighbor binary co-occurrence, three groups of 59 local binary patterns are extracted from the spatial domain of red and green color channels, together with their corresponding prediction errors. The 1st-level diagonal

wavelet sub-bands of the prediction errors are extracted and used as features at the input of a multi-class SVM.

In [69] Milani et al. proposed the idea of eigenalgorithms to exploit the CFA interpolation differences for camera model identification. The idea behind the eigenalgorithm is to re-interpolate the image with a set of distinctive strategies, then correlate the obtained re-interpolated images with the original one. The image under analyses is the characterized in a "space of strategies". This representation is used as a feature vector to feed an SVM classifier.

Thai et al. [70] design a statistical test for the camera model identification problem. The approach is based on a noise model for natural raw image characterized by only two parameters, which are considered as unique fingerprint to identify camera models. The problem is cast in the framework of hypothesis testing theory. Two generalized Likelihood Ratio Tests are designed to deal with the unknown model parameters

The introduction of rich models for steganalysis [71] based on co-occurrences inspired the authors of [72–74] to define a series of low-level features extracted from noise-based signals computed starting from the image under investigation.

Chen et al. [72] build a rich model on top of a series of 4 camera's demosaicing algorithms. An image is demosaiced based on the 4 models and the demosaicing error is computed for each demosaicing strategy. For every demosaicing error, 343 co-occurrences are computed with quantization and truncation as from [71]. A random forest of binary classifiers is trained on top of the 1372 features, where each base learner (Fisher Linear Discriminator) learns from a subset of training samples and features. Similarly, Marra et al. [73] exploit horizontal and vertical high-pass filtered versions of the image to propose a set of co-occurrences then selected via Principal Components Analysis. The projected features are fed to an SVM classifier. Tuama et al. [74] computed co-occurrences from the PRNU residual extracted from the image. Together with color dependencies and conditional probabilities across color channels of the extracted PRNU residual a total of 72 features is collected and fed to an SVM classifier.

A more comprehensive overview over camera model identification systems can be found in [29].

2.3.2 Tampering detection and localization

Image tampering detection and localization techniques have been developed over time by a number of researchers focusing on copy-move forgeries, splice forgeries, inpainting, image-wise adjustments (resizing, histogram equalization, cropping, etc.) and many more. Following the survey presented by Zampoglou et al. [75], we provide an overview

of the literature methods against which our algorithm will be compared. Specifically, we focus on algorithms that do not need any prior information (e.g. JPEG header, additional inputs such as reference PRNUs, etc.), and follow the naming convention used in [75] and the derived toolbox.

ADQ1 Aligned Double Quantization detection, developed by Lin et al. in [76] aims at discovering the absence of double JPEG compression traces in tampered 8×8 image blocks. Posterior probabilities for each block being possibly tampered are generated through voting among discrete cosine transform (DCT) coefficients histograms collected for all blocks in the image.

BLK Due to its nature, JPEG compression introduces 8×8 periodic artifacts on images that can be highlighted thanks to Block Artifact Grids (BAG) technique from Li et al. [77]. Detecting disturbances in the BAG allows to find traces of image cropping (grid misalignment), painted regions, copy-move portions.

CFA1 At acquisition time, color images undergo Color Filter Array (CFA) interpolation due to the nature of acquisition process. Detecting anomalies in the statistics of CFA interpolation patters allows Ferrara et al. [78] to build a tampering localization system. A mixture of Gaussian distributions is estimated from all the 2×2 blocks of the image, resulting in a fine-grained tamper probability map.

CFA2 By estimating CFA patterns within four common Bayer patterns, Dirik and Memon [79] show that if none of the candidate patterns fits sufficiently better than the other in a neighborhood, it is likely that tampering occurred. In fact, weak traces of interpolation left by the de-mosaicing algorithm are hidden by typical splicing operations like resizing and rotations.

DCT Inconsistencies in JPEG DCT coefficients histograms are detected in [80] by first estimating the quantization matrix from trusted image blocks, then evaluating suspicious areas with a blocking artifact measure (BAM).

ELA Error Level Analysis [81] aims to detect parts of the image that have undergone fewer JPEG compressions than the rest of the image. It works by intentionally re-saving the image at a known error rate and then computing the difference between the original and the re-compressed image. Local minima in image difference indicate original regions, whereas local maximums are symptoms of image tampering.

GHO JPEG Ghosts detection by Farid [82] is an effective technique to identify parts of the image that were previously compressed with smaller quality factors. The method is based on finding local minimum in the sum of squared differences between the image under analysis and its re-compressed version with varying quality factors.

NOI1 Mahdian et al. [83] build their tampering detector upon the hypothesis that

usually Gaussian noise is added to tampered regions with the goal of deceiving classical image tampering detection algorithms. Modeling the local image noise variance through wavelet filtering, a segmentation method based on homogeneous noise levels is used to provide an estimate of the tampered regions within an image.

NOI4 A statistical framework aimed at fusing several sources of information about image tampering is presented by Fontani et al. [84]. Information provided by different forensic tools yields to a global judgment about the authenticity of an image. Outcomes from several JPEG-based algorithms are modeled and fused using Dempster–Shafer Theory of Evidence, being capable of handling uncertain answers and lack of knowledge about prior probabilities.

Among the works on tampering detection and localization, Cozzolino et al. [85] propose a method based on discretized high frequency components statistics to localize tampering. Local co-occurrence map of quantized and truncated high-frequency components of the image are used to fit a two-class model via Expectation-Maximization. The base assumption is that spliced area will have different characteristics from the rest of the image. The method is further refined in [86], where an auto-encoder network is used to learn the characteristics of the background in order to highlight, as reconstruction errors, the portions of the image that have been spliced. In [87] the same authors leverage the co-occurrences of high-frequency residuals to train a Long Short-Term Memory (LSTM) neural network capable of detecting video forgeries due to the insertion of objects from different scenes.

2.3.3 Antiforensics

As most of the camera model identification systems from the literature Section 2.3.1 are based on demosaicing artifacts, the antiforensics methods developed over the last 10 years are as well based on mis-leading CFA recognition or demosaicing algorithms identification.

Chuang et al. [88] proposed two anti-forensic techniques to disguise camera model identification based on color interpolation artifacts. A first technique employs parameter perturbation in order to mislead the identification with targeted interpolation algorithm. The main idea is to prevent a color-interpolation-based identification system from identifying a specific interpolation algorithm. The second technique deals with algorithm mixing, showing that it is also possible to modify an image so that it is identified as being interpolated with a interpolation algorithm than the one used by the camera firmware.

Kirchner et al. [89] introduced a CFA synthesis method based on the idea of re-interpolation of the color planes while minimizing the distortion between the input image

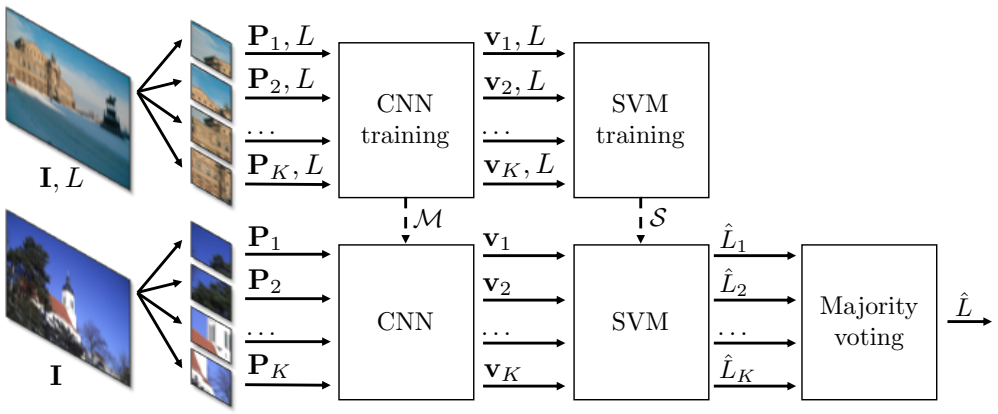


Figure 2.3: Proposed pipeline for camera model attribution. (Top) CNN and SVM training process: patches extracted from each training image $\boldsymbol{I}$ inherit the same label L of the image. (Bottom) Evaluation pipeline: for each patch $\boldsymbol{P}_k$ from the image $\boldsymbol{I}$ under analysis, a feature vector $\boldsymbol{b}_k$ is extracted through the CNN. Feature vectors are fed to a set of linear SVM classifiers in order to associate a candidate label $\hat{L}_k$ to each vector. The predicted label $\hat{L}$ for image $\boldsymbol{I}$ is obtained by majority voting.

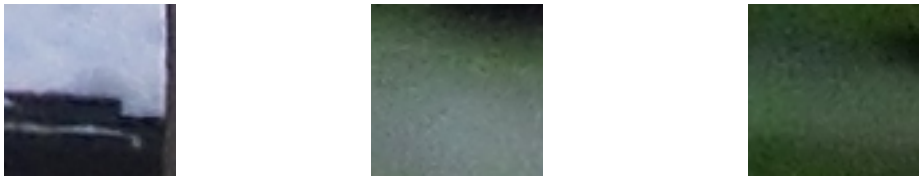


Figure 2.4: Patch examples and with quality measures $0.91, 0.78, 0.54$, left to right

and the output image. This method easily fools interpolation-based interpolation detectors as the one presented in [58]. Moreover is is not trivial to detect the presence of such an antiforensics attack on the anonymized image.

2.4 Camera Model Identification

The problem of camera model identification consists in detecting the model L (within a set of known camera models) used to shoot the image $\boldsymbol{I}$, a closed set classification problem. In the following we describe the proposed algorithm to solve this problem. We first explain how to perform the training step needed to learn the CNN and SVMs parameters. Then, we report how to use the trained algorithm for classifying new images under analysis. Figure 2.3 represents in a schematic way both training and evaluation pipelines.

2.4.1 Patch selection

For each color image $\boldsymbol{I}$, associated to a specific camera model L, we extract K non overlapping patches $\boldsymbol{P}_k$, $k \in [1, K]$, of size 64×64 pixels. The rationale behind this choice is twofold: (i) splitting images into patches allows us to obtain a greater amount of data for CNN training; (ii) feeding the CNN with smaller data (i.e., a patches rather than full resolution images) enables working with smaller and lighter CNN architectures. In order to avoid selecting overly dark or saturated regions, we exclude all patches with saturated pixels and prioritize those whose average value is close to half the image dynamic. To this end, we devised a patch selection procedure. Specifically, for each patch $\boldsymbol{P}_k$ within an image, we compute a quality value defined as

$$Q(\boldsymbol{P}_k) = \frac{1}{3} \sum_{c \in [R,G,B]} \left[\alpha \cdot \beta \left(\mu_{\mathbf{c}} - {\mu_{\mathbf{c}}}^2 \right) + (1 - \alpha) \left(1 - e^{\gamma \sigma_{\mathbf{c}}} \right) \right], \tag{2.3}$$

where α, β and γ are empirically set constants (set to 0.7, 4 and $\ln(0.01)$ in our experiments), whereas μ_c and σ_c, $c \in [R, G, B]$ are the average and standard deviation of red, green and blue components (in range $[0, 1]$) of patch $\boldsymbol{P}_k$, respectively. This quality measure tends to be lower for overly saturated or flat patches, whereas it is higher for textured patches showing some statistical variance (as shown in Figure 2.4). Therefore, we select, for each image, the K patches with the highest Q values. This way, saturated patches will not be considered during either training, validation or testing.

2.4.2 CNN Training

Given a set of training labeled images coming from N known camera models, we split them into patches and associate to each patch the same label L of the source image to which patches belong. Then, we feed the CNN with all available patch-label pairs for training. The choice of the CNN architecture is a delicate step. As an example, a too deep network may be unnecessary long to train and may contain too many parameters that need a huge training dataset to be properly tuned. However, smaller networks may not achieve accurate enough performance, loosing the ability of well discriminating the used camera models. Moreover, despite the number of used layers, also the choice of filters size and stride plays a crucial role, not to mention the use of fully-connected layers. For these reasons we tested different CNN architectures, varying for each architecture the number of layers and the hyper-parameters.

A common aspect among all tested architectures is that they accept as input patches of size $64 \times 64 \times 3$, with pixel values between 0 and 255. The pixel-wise average over

the training set is subtracted to each input patch. The result is then scaled pixel-wise in amplitude by a factor of 0.0125 to reduce its dynamic. At the end of the training step, we obtain the CNN model $\mathcal{M}$. As trained CNN model $\mathcal{M}$, we select the one that provides the smallest loss on validation patches within the first 50 training epochs.

Designing a proper CNN architecture for our camera model identification pipeline is a critical step. The overall accuracy of the system is significantly determined by the extracted feature vectors from each image patch. There are several key design choices that have to be considered as they determine the final structure of the CNN. The depth of the network, the use of pooling layers and the size of the kernels are examples and are referred to as hyper-parameters. Tuning hyper-parameters is approached in a trial-and-error fashion as there are no hard-quantitative rules that can be followed. This is due to the fact that we approach camera model identification as a data-driven problem and the final architecture of the network depends on the type of data under consideration. In our particular case, we explore networks that accept input patches of size $64 \times 64 \times 3$. By using some of the CNN architecture design guidelines proposed in [90] we proposed a baseline CNN architecture, hereinafter denoted as $\mathcal{M}_{Conv4}$, defined as follows:

- An RGB color input patch of size 64×64 is fed as input to the first Convolutional layer with kernel size $4 \times 4 \times 3$ producing 32 feature maps as output. Filtering is applied with stride 1.
- The resulting $63 \times 63 \times 32$ feature maps are aggregated with a Max-Pooling layer with kernel size 2×2 applied with stride 2, producing $32 \times 32 \times 32$ feature maps.
- A second Convolutional layer with 48 filters of size $5 \times 5 \times 32$ applied with stride 1 generates $28 \times 28 \times 48$ feature maps.
- A Max-Pooling layer with kernel size 2×2 applied with stride 2 produces a $14 \times 14 \times 48$ feature maps.
- A third Convolutional layer with 64 filters of size $5 \times 5 \times 48$ applied with stride 1 generates $10 \times 10 \times 64$ feature maps.
- A Max-Pooling layer with kernel size 2×2 applied with stride 2 produces a $5 \times 5 \times 64$ feature map.
- A fourth Convolutional layer with 128 filters of size $5 \times 5 \times 64$ applied with stride 1 generates a vector of 128 elements.
- A fully-connected layer with 128 output neurons followed by a ReLU layer produces the 128 element feature vector.

- A last fully-connected layer with N_{cams} output neurons followed by a Softmax layer acts as logistic regression classifier during CNN training phase. N_{cams} is the number of camera model used at CNN training stage.

The overall architecture is characterized by $340,462$ parameters, learned through Stochastic Gradient Descent on batches of 128 patches. In Table 2.1 we summarize its hyper-parameters.

In Section 2.7, we study the behavior of $\mathcal{M}_{Conv4}$ compared to other 3 new networks, which vary in depth. In particular, we consider $\mathcal{M}_{Conv4}$ as the base CNN on top of which we develop the 3 remaining deeper architectures. As shown by the Oxford VGG team in [91] and Szegedy et al. in [37] the representational capacity of a network is largely determined by its depth. This insight was also verified with the recent development of deep residual networks [92], which can have more than 150 layers and were used by the winners of the ILSVRC-2015 competition [93]. Keeping simplicity in mind, and following the strategy that Simonyan et al. devised in [91] to prepare their submission for the ILSVRC-2014, we explored a single family of networks of increasing depth. By doing so, we took advantage of the key design choices made in our base CNN model, $\mathcal{M}_{Conv4}$, such as the stacks of convolutional and pooling layers structure and the kernel sizes of the convolutional layers. A stride value of 1 was chosen for the convolutional layers. This results in no skipping (i.e. our filters are applied to all the values of the input volumes they receive).

The CNN architectures evaluated in Section 2.7 are outlined in Table 2.2, one per column. The baseline architecture $\mathcal{M}_{Conv4}$ was introduced earlier, and we refer to the three remaining networks as $\mathcal{M}_{Conv6}$, $\mathcal{M}_{Conv8}$ and$\mathcal{M}_{Conv10}$. As already mentioned, the 4 networks vary only in the depth: from 6 weight layers in the baseline network $\mathcal{M}_{Conv4}$ (4 convolutional and 2 fully-connected layers) to 12 weight layers in the network $\mathcal{M}_{Conv10}$ (10 convolutional and 2 fully-connected layers). The number of filters of each convolutional layer is rather small, starting from 32 in the first layer and then adding 16 more filters after each pooling layer, except for the last one, where we increase the number of filters by a factor of 2 reaching a total of 128 filters.

2.4.3 SVM Training

Even though we could perform classification by simply thresholding the output of the last network layer, we decided to make use of an additional classification tool. For each patch, the selected CNN model $\mathcal{M}$ is used to extract a feature vector of 128 elements, stopping the forward propagation at the *ReLU-1* layer. Feature vectors associated to

Table 2.1: Structure of the reference CNN architecture $\mathcal{M}_{Conv4}$. N is the number of training classes. Feature are extracted after the ReLU-1 layer.

Layer	**Input size**	**Kernel size**	**Stride**	**Num filters**	**Output size**
Conv-1	$64 \times 64 \times 3$	4×4	1	32	$61 \times 61 \times 32$
Pool-1	$61 \times 61 \times 32$	2×2	2	-	$31 \times 31 \times 32$
Conv-2	$31 \times 31 \times 32$	5×5	1	48	$27 \times 27 \times 48$
Pool-2	$27 \times 27 \times 48$	2×2	2	-	$14 \times 14 \times 48$
Conv-3	$14 \times 14 \times 48$	5×5	1	64	$9 \times 9 \times 64$
Pool-3	$9 \times 9 \times 64$	2×2	2	-	$5 \times 5 \times 64$
Conv-4	$5 \times 5 \times 64$	5×5	1	128	$1 \times 1 \times 128$
FullyConnected-1	$1 \times 1 \times 128$	-	-	128	28
ReLU-1	128	-	-	-	128
FullyConnected-2	128	-	-	N	N
SoftMax	N	-	-	-	N

Table 2.2: The 4 proposed CNN architectures (shown in columns). Added layers are shown in bold and the number of filters for each convolutional layer is shown in parenthesis

CNN Architecture			
$\mathcal{M}_{Conv4}$	$\mathcal{M}_{Conv6}$	$\mathcal{M}_{Conv8}$	$\mathcal{M}_{Conv10}$
6 weight layers	8 weight layers	10 weight layers	12 weight layers
Input ($64 \times 64 \times 3$ image patch)			
Conv-1 (32)	Conv-1 (32) **Conv-1** (32)	Conv-1 (32) Conv-1 (32)	Conv-1 (32) Conv-1 (32)
Pool-1			
Conv-2 (48)	Conv-2 (48) **Conv-2** (48)	Conv-2 (48) Conv-2 (48)	Conv-2 (48) Conv-2 (48)
Pool-2			
Conv-3 (64)	Conv-3 (64)	Conv-3 (64) **Conv-3** (64)	Conv-3 (64) Conv-3 (64) **Conv-3** (64)
Pool-3			
Conv-4 (128)	Conv-4 (128)	Conv-4 (128) **Conv-4** (128)	Conv-4 (128) Conv-4 (128) **Conv-4** (128)
FullyConnected-1			
ReLU-1			
FullyConnected-2			
SoftMax			

training patches are used to train a set of $N \cdot (N-1)/2$ linear binary SVM classifiers $\mathcal{S}$ in a One-versus-One fashion. The regularization constraint C is selected to maximize

classification accuracy on validation patches.

2.4.4 Majority Voting

When a new image $\boldsymbol{I}$ is under analysis, the camera model used to shoot it is estimated as follows. A set of K patches is obtained from image $\boldsymbol{I}$ according to the quality measures described above. Each patch $\boldsymbol{P}_k$ is processed by CNN model $\mathcal{M}$ in order to extract a feature vector $\boldsymbol{v}_k$. The set $\mathcal{S}$ of linear SVMs assigns a label $\hat{L}_k$ to each patch by attributing vectors $\boldsymbol{v}_k$ to one of the available N classes (i.e., camera models). The predicted model $\hat{L}$ for image $\boldsymbol{I}$ is obtained through majority voting on $\hat{L}_k, k \in [1, K]$. In case of par, random selection between equally likely models is operated.

2.4.5 Training Strategies

As discussed in [29], the training procedure for machine learning-based camera model identification algorithms must be devised with great attention. As a matter of fact, on one hand it is important to ensure a sufficient number of training data. On the other hand, training data cannot be just randomly selected. Instead, they must be carefully chosen in order to avoid over-fitting and ensure a wide variety of images covering different scenarios.

In order to further highlight the importance of the training strategy, let us consider the following example. Let us consider camera model identification problem using only two camera models whose labels are L_1 and L_2, respectively. If all images coming from camera L_1 are very dark, and all images from camera L_2 are very bright, the CNN might learn to discriminate luminance levels rather than camera models. It is clear that, in order to avoid such a biased training inevitably leading to incorrect results and conclusions, images from both cameras must depict both dark and bright scenes in this case. Despite the simplicity of this example, the situation becomes less trivial when many different camera models and images with different semantic contents are used.

In order to consider this important issue, in our study we consider the Dresden Image Dataset [94] as reference, as suggested in [29]. This dataset is composed by 73 camera devices from 25 camera models from 14 manufacturers. For each device a variable number of shots has been taken in several geographical positions. For each position a set of different motives is shot. Details about the acquisition process are available at [94]. In the following we will refer to *scene* when considering the combination of a geographical position with a specific motive. This results in a total amount of 83 available scenes. We only consider camera models represented by more than one device, in order to ensure that

the CNN learns model specific artifacts rather than instance specific ones. This leads to a dataset composed of 18 camera models (as Nikon D70 and D70s basically differ only in their on-device screen), for nearly $15,000$ shots.

For our problem, we need to split images in three different datasets: (i) a training set $\mathbb{D}_T$ used for updating CNNs and SVMs parameters; (ii) a validation set $\mathbb{D}_V$ used to decide the stopping point of the training step and avoid over-fitting (i.e., typically the training process is stopped when validation loss, given by SoftMax layer, reaches its minimum); (iii) an evaluation set $\mathbb{D}_E$ used to test the trained architectures. Following the ideas presented by Kirchner et al. [29]

- we selected shots belonging to the evaluation set ($\mathbb{D}_E$) from N_E scenes and a single instance per camera model. The selected images are never used in training or validation.

- we selected shots for training set ($\mathbb{D}_T$) and validation set ($\mathbb{D}_V$) among images from remaining scenes and instances.

Specifically, we define three splitting policies for $\mathbb{D}_T$ and $\mathbb{D}_V$ so to test for possible over-fitting on scenes content rather than on camera model identification during CNN training. Since the validation set is used to decide when to stop the training process, if its content is too similar to the training set we could easily over-fit. Conversely, if the validation set is sufficiently different from the training set, we should be able to obtain more generalizable results on the evaluation set. Splitting policies are detailed below:

1. *Fair-N_T*: training and validation shots are split according to the depicted scene. The number of training scenes is set to N_T and shots coming from a specific scene are included only in $\mathbb{D}_T$ or in $\mathbb{D}_V$. In this way, $\mathbb{D}_T$ and $\mathbb{D}_V$ are completely disjoint sets (in terms of scenes), thus should lead to robust training.

2. *Fair-balanced-N_T*: training and validation shots are split according to scenes as for *Fair-N_T*. The number of shots for each device model is the same, leading to a model-balanced training dataset.

3. *Unfair-P_T*: training and validation shots are split regardless of the scene they belong to, fixing the percentage of training shots to P_T. In doing so, the same scene can appear in both training and validation sets, thus possibly leading to over-fitting and less accurate evaluation results.

A small case example for the three splitting strategies is available at Table 2.3.

Table 2.3: A small scale example for three different splitting strategies. Row colors correspond to scenes. First, an instance id and a set of scenes are selected for the evaluation set $\mathbb{D}_E$. Considering the remaining instances and scenes, $\mathbb{D}_T$ and $\mathbb{D}_V$ are built according to what specified in the text. Labels E, V and T denote images associated to $\mathbb{D}_E$, $\mathbb{D}_V$ and $\mathbb{D}_T$ according to each policy.

Brand	Model	Instance	Scene	*Fair*	*Fair-balanced*	*Unfair*
Canon	Ixus 70	0	Kohlenstrasse Back view ZINT	E	E	E
Canon	Ixus 70	0	Home III Trees in a garden II			
Canon	Ixus 70	0	Kaethe-Kollwitz-Ufer Blue Wonder			
Canon	Ixus 70	1	Kohlenstrasse Back view ZINT			
Canon	Ixus 70	1	Home III Trees in a garden II	V	V	T
Canon	Ixus 70	1	Kaethe-Kollwitz-Ufer Blue Wonder	T	T	T
Canon	Ixus 70	1	Kaethe-Kollwitz-Ufer Blue Wonder	T		V
Kodak	M1063	0	Kohlenstrasse Back view ZINT	E	E	E
Kodak	M1063	0	Home III Trees in a garden II			
Kodak	M1063	0	Kaethe-Kollwitz-Ufer Blue Wonder			
Kodak	M1063	1	Kohlenstrasse Back view ZINT			
Kodak	M1063	1	Home III Trees in a garden II	V	V	V
Kodak	M1063	1	Kaethe-Kollwitz-Ufer Blue Wonder	T	T	T

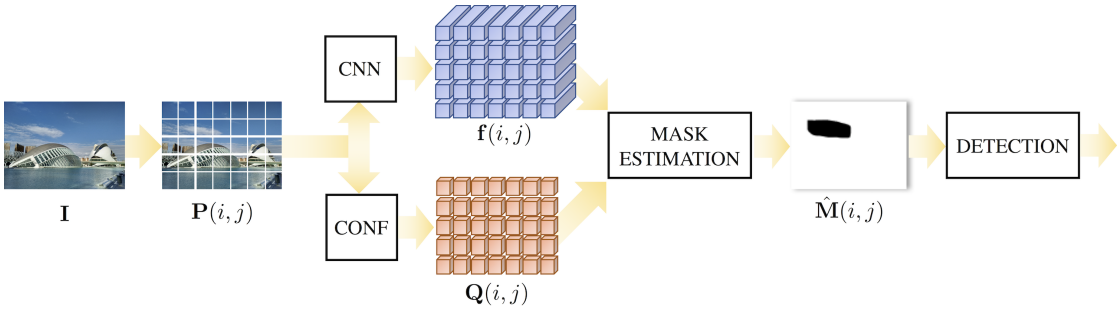


Figure 2.5: Pipeline of the proposed method. An image $\boldsymbol{I}$ is split into patches $\mathbf{P}(i,j)$. Each patch is described by a feature vector $\mathbf{f}(i,j)$ extracted through a CNN, and a confidence score $\mathbf{Q}(i,j)$. A custom clustering algorithm produces a tampering mask prediction $\hat{\mathbf{M}}$, which is also used for detection.

2.5 Tampering detection and localization

In this section we present the proposed method for image forgery detection and localization in case of images generated through composition of pictures shot with different camera models. In this scenario, we consider that pristine images are pictures directly obtained from a camera. Conversely, forged images are those created taking a pristine

image, and pasting onto it one or more small regions taken from pictures shot with different camera models with respect to the background shot. Under these assumptions, the proposed method is devised to estimate whether the totality of image patches come from a single camera (i.e. the image is pristine), or some portions of the image are not coherent with the rest of the picture in terms of camera attribution (i.e. the image is forged). If this is the case, we also localize the forged region.

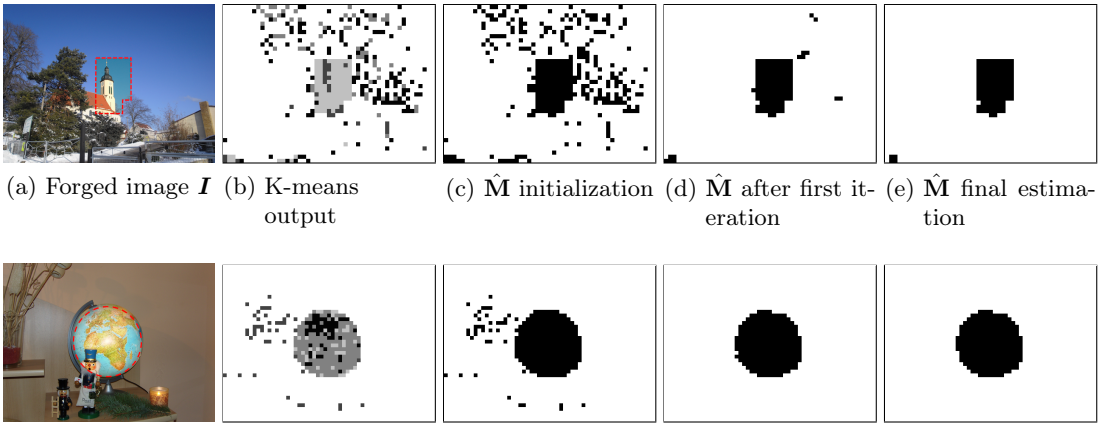
Figure 2.5 outlines the proposed pipeline used to detect and localize tampered regions within images. An image $\boldsymbol{I}$ is first divided into non-overlapping patches. Each patch $\mathbf{P}$ is fed as input to a pre-trained CNN to extract a feature vector $\mathbf{f}$ of N_{cams} elements. Information about patch position, feature vectors $\mathbf{f}$ and patch confidence are given as input to the clustering algorithm that estimates a tampering mask. The final output $\hat{\mathbf{M}}$ is a binary mask, where 0s indicate patches belonging to the pristine region and 1s indicate forged patches. If no (or just a few) forged pixels are detected, the image is considered pristine. In the following, we report a detailed explanation of each algorithmic step.

2.5.1 Feature Extraction

The first step of our algorithm consists in splitting an image into 64×64 color patches, and extracting a feature vector containing camera model information from each one of them.

Formally, let us define with $I_{x,y}$, $x \in [1, N_x]$, $y \in [1, N_y]$ the pixel in position (x, y) of the image $\boldsymbol{I}$ under analysis. Similarly, let us define 64×64 patches as $\boldsymbol{P}(i, j)$, $i \in [1, N_i], j \in [1, N_j]$, where $N_i = \frac{N_x}{64}$ and $N_j = \frac{N_y}{64}$ are the numbers of patches per column and row, respectively. In other words a patch $\mathbf{P}(i, j)$ corresponds to pixels $I_{x,y}$, $x \in [64(i-1)+1, 64 \cdot i]$, $y \in [64(j-1)+1, 64 \cdot j]$. Each patch is fed to the pre-trained CNN $\mathcal{M}_{Conv4}$ presented in Section 2.4, which outputs a N_{cams}-dimensional feature vector $\mathbf{f}(i, j)$ after its Softmax layer.

Notice that features $\mathbf{f}(i, j)$ are vectors whose elements are positive and add to 1. Ideally, if a patch comes from a camera model used for CNN training, $\mathbf{f}(i, j)$ should present a maximum close to 1 in a single position indicating the used camera model. However, in case of cameras never seen by the network, or simply due to noise, $\mathbf{f}(i, j)$ may present different behaviors. However, as will be shown in Section 2.7, this feature vector is capable of extracting camera model information that generalizes to models never used in training. Therefore, we expect that $\mathbf{f}(i, j)$ for patches coming from a single camera are coherent and can be clustered together in the feature space. This enables localization of pixel regions coming from different devices, even if unknown.



(a) Forged image $\boldsymbol{I}$ (b) K-means output (c) $\hat{\mathbf{M}}$ initialization (d) $\hat{\mathbf{M}}$ after first iteration (e) $\hat{\mathbf{M}}$ final estimation

(f) Forged image $\boldsymbol{I}$ (g) K-means output (h) $\hat{\mathbf{M}}$ initialization (i) $\hat{\mathbf{M}}$ after first iteration (j) $\hat{\mathbf{M}}$ final estimation

Figure 2.6: Example of forged images and intermediate outputs of the proposed localization algorithm. Forgeries are highlighted in red dashed lines. K-means detected clusters are mapped to different colors. White and black pixels represent 0 and 1 values of $\hat{\mathbf{M}}$, respectively.

2.5.2 Confidence Computation

As discussed in Section 2.4 the output of the used CNN is expected to be more reliable on some image patches than others. Indeed, not all patches contain enough statistical information about the used camera model. Therefore, we associate a confidence value $Q_{i,j}$ to each patch as defined in Equation (2.3). Notice that $Q_{i,j} \in [0, 1]$ for convenience. The lower the value, the less confident the algorithm is about the patch.

2.5.3 Tampering Mask Estimation

Once we obtain confidence $Q_{i,j}$ and feature $\mathbf{f}(i, j)$ for each patch $\mathbf{P}(i, j)$, we make use of this information to estimate a tampering mask $\hat{M}_{i,j}$. This mask is a binary matrix, where $\hat{M}_{i,j} = 0$ indicates that $\mathbf{P}(i, j)$ is pristine, whereas $\hat{M}_{i,j} = 1$ indicates that $\mathbf{P}(i, j)$ is an alien patch.

To initialize $\hat{\mathbf{M}}$, we assume that the majority of patches comes from a single camera, whereas only spliced regions come from different camera models. Therefore, the majority of vectors $\mathbf{f}(i, j)$ should be coherent (i.e. those belonging to the pristine region), whereas other features should group into different clusters (e.g. due to noise, low confidence, or because they belong to alien regions). We therefore apply K-means clustering algorithm to features $\mathbf{f}(i, j)$ to perform a first rough detection of which patches belong to the pristine portion of the image. In order to be robust in this initialization step, we set the initial number of clusters $N_{\text{clusters}} = 5$ (i.e. greater than the expected number of cameras used

Algorithm 1 Tampering mask estimation

Require: $\mathbf{f}(i,j)\ i \in [1, N_i],\ j \in [1, N_j]$ ▷ Feature vectors
Require: $Q_{i,j}\ i \in [1, N_i],\ j \in [1, N_j]$ ▷ Confidence
Require: $\Gamma_{\text{dist}}, \Gamma_{\text{conf}}$ ▷ Thresholds
Require: N_{clusters} ▷ Number of clusters

function EstimateMask

cluster$(i,j) \leftarrow$ Kmeans$(\{\mathbf{f}(i,j)\}_{\forall(i,j)}, N_{\text{clusters}})$ ▷ Mask initialization
for $i \in [1, N_i],\ j \in [1, N_j]$ **do**
if cluster(i,j) is the largest one **then**
$\hat{M}_{i,j} \leftarrow 0$
else
$\hat{M}_{i,j} \leftarrow 1$
end if
end for

repeat ▷ Refinement
$\bar{\mathbf{f}} \leftarrow$ average$(\{\mathbf{f}(i,j)\}_{(i,j)\,|\,\hat{M}_{i,j}=0})$ ▷ Centroid estimation

for $(i,j)\,|\,\hat{M}_{i,j} = 1$ **do** ▷ Feature space refinement
$d(i,j) \leftarrow \frac{\sum_{k=1}^{N_{\text{cams}}} |\bar{\mathbf{f}}(i,j)_k - \mathbf{f}(i,j)_k|}{\sum_{k=1}^{N_{\text{cams}}} (\bar{\mathbf{f}}(i,j)_k + \mathbf{f}(i,j)_k)}$
if $d(i,j) < \Gamma_{\text{dist}}$ **then** $\hat{M}_{i,j} = 0$
$\hat{M}_{i,j} \leftarrow 0$
end if
end for
$\hat{\mathbf{M}} \leftarrow$opening$(\hat{\mathbf{M}})$ ▷ Geometric space refinement
until $\hat{\mathbf{M}}$ changes with respect to the previous iteration

for $i \in [1, N_i],\ j \in [1, N_j]$ **do** ▷ Confidence thresholding
if $Q_{i,j} < \Gamma_{\text{conf}}$ **then**
$\hat{M}_{i,j} \leftarrow 0$
end if
end for
return $\hat{\mathbf{M}}$
end function

for the forgery). We then set $\hat{M}_{i,j} = 0$ if $\mathbf{f}(i,j)$ belongs to the cluster with the highest cardinality (i.e. pristine region). Conversely, we set $\hat{M}_{i,j} = 1$ if $\mathbf{f}(i,j)$ belongs to any other cluster (i.e. possibly forged region). An example of this step output is shown for

two forged images in Figure 2.6.

After mask initialization, we need to refine our estimate about all patches initialized as forged. To this purpose, we apply the following iterative procedure:

1. We compute the centroid $\bar{\mathbf{f}}$ of all features $\mathbf{f}(i,j)$ for which $\hat{M}_{i,j} = 0$ (i.e. pristine ones). Formally,
$$\bar{\mathbf{f}} = \text{average}(\{\mathbf{f}(i,j)\}_{(i,j)\,|\,\hat{M}_{i,j}=0}), \tag{2.4}$$
where average$(\cdot)$ computes the average vector in the set.

2. We compute the Bray-Curtis distance between $\bar{\mathbf{f}}$ and each $\mathbf{f}(i,j)$ for which $\hat{M}_{i,j} = 1$ defined as
$$d(i,j) = \frac{\sum_{k=1}^{N_{\text{cams}}} |\bar{\mathbf{f}}(i,j)_k - \mathbf{f}(i,j)_k|}{\sum_{k=1}^{N_{\text{cams}}} (\bar{\mathbf{f}}(i,j)_k + \mathbf{f}(i,j)_k)}, \tag{2.5}$$
where $\mathbf{f}(i,j)_k$ is the k-th element of vector $\mathbf{f}(i,j)$. Notice that all considered vectors add to 1, thus $d(i,j) \in [0,1]$.

3. We refine pristine region in the feature space by considering as pristine all patches with feature vector close to the pristine centroid $\bar{\mathbf{f}}$. Formally,
$$\hat{M}_{i,j} = 0 \quad \text{if} \quad d(i,j) < \Gamma_{\text{dist}}, \tag{2.6}$$
where Γ_{dist} is a threshold selected in range $[0,1]$ balancing probability of true positive and true negative detections, as shall be explained in the experimental section.

4. We finally refine pristine region estimate in the geometric space, by aggregating to pristine region spurious isolated patches considered forged. The idea is that each forgery should not be smaller than a given pixel size. Formally, we achieve this goal using opening morphological operator to $\hat{\mathbf{M}}$ with a 2×2 structuring element of ones. Notice that this means we consider that the smallest possible forgery is a 128×128 pixel region.

This procedure is iterated until convergence (i.e. all patches are identified as pristine, or $\hat{\mathbf{M}}$ estimate does not change with respect to previous step).

After last iteration, we take into account feature confidence for each patch, i.e. $Q_{i,j}$. Specifically, we decide to be conservative and set as pristine all patches for which we are not confident enough about their camera model estimation. Formally,
$$\hat{M}_{i,j} = 0 \quad \text{if} \quad Q_{i,j} < \Gamma_{\text{conf}}, \tag{2.7}$$

where Γ_{conf} is a threshold selected in range $[0, 1]$ (i.e. 0 means we do not take confidence into account, 1 sets all patches as pristine). For the sake of clarity, the pseudo-code is reported in Algorithm 1. Figure 2.6 shows an example of estimated masks for two forged images.

2.5.4 Tampering Detection

Once mask $\hat{\mathbf{M}}$ has been estimated, we decide whether image $\boldsymbol{I}$ is pristine or not based on the amount of estimated forged pixels. Formally,

$$\begin{cases} \boldsymbol{I} \quad \text{is pristine} & \text{if } \mu_{\hat{\mathbf{M}}} \leq \Gamma_{\text{det}}, \\ \boldsymbol{I} \quad \text{is forged} & \text{if } \mu_{\hat{\mathbf{M}}} > \Gamma_{\text{det}}, \end{cases} \tag{2.8}$$

where $\mu_{\hat{\mathbf{M}}}$ is the average value of $\hat{\mathbf{M}}$ which spans the range $[0, 1]$, and Γ_{det} is a threshold indicating how many pixels we need to identify as forged to estimate that the image is actually forged (i.e. 0 means that images are considered pristine only if all patches are detected as pristine, 1 means that images are always considered pristine).

2.6 Antiforensics

In this section we describe the antiforensic approach used to fool a CNN trained to perform camera model identification. Figure 2.7 shows the block diagram of our proposed counter-forensic method. Our method consists of an adversarial image generator module that can be added to a CNN-based camera model evaluation pipeline. In Figure 2.7, we assume a similar structure to the previously presented pipeline in Section 2.4. Our adversarial image generator module takes as input the set of K patches that have been extracted from the image $\boldsymbol{I}$ that is being analyzed. When presented with new image patches, our module can work in two different modes.

In the first operation mode, the adversarial image generator module does an untargeted image manipulation, that is, it does not try to perturb the image patches to produce a specific misclassification class. Instead, we use the derivative of the loss function of the CNN with respect to the input image patches to add a perturbation to the images. The derivative is computed using backpropagation with the labels $\hat{L}'_k$, $k \in [1, K]$ that are given by the CNN detector when it first processes the unmodified image patches. This procedure is known as the fast gradient sign method (FGSM), see Section 2.2.4 .

In the second operation mode, the adversarial image generator module does a targeted image manipulation. In this case, we try to perturb the image patches to produce a

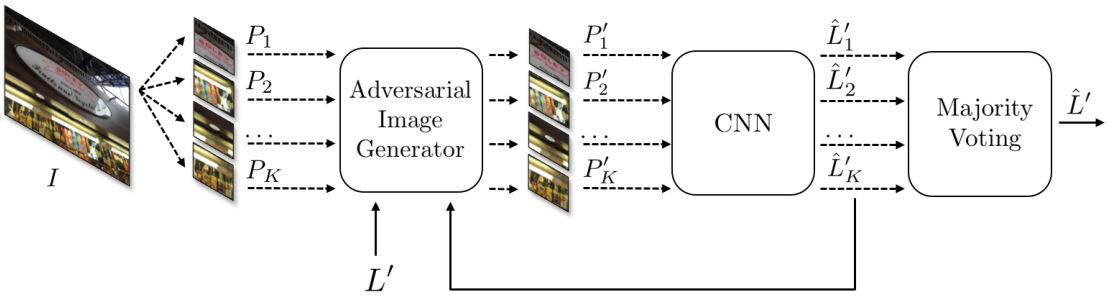


Figure 2.7: Block diagram of our proposed method.

specific misclassification class L', different from the true real label L that is associated with the analyzed image $\boldsymbol{I}$ and its associated $\boldsymbol{P}_k$ patches. In this mode of operation, we exploit the forward derivative of a CNN to find an adversarial perturbation that will force the network to misclassify the image patch into the target class by computing the adversarial saliency map. Starting with an unmodified image patch, we perturb each feature by a constant offset ϵ. This process is repeated iteratively until the target misclassification is achieved. This procedure is known as the Jacobian-based saliency map attack (JSMA), see Section 2.2.4.

2.6.1 Implementation Details

To implement our counter-forensic method, we have used the software library *cleverhans* [95]. The library provides standardized reference implementations of adversarial image generation techniques and adversarial training. The library can be used to develop more robust CNN architectures and to provide standardized benchmarks of CNNs performance in an adversarial setting. As noted in [95], benchmarks constructed without a standardized implementation of adversarial image generation techniques are not comparable to each other, because a good result may indicate a robust CNN or it may merely indicate a weak implementation of the adversarial image generation procedure.

2.7 Experiments

In the following we show the results obtained in terms of camera model identification, tampering detection and localization and counter-forensics attacks to CNN for camera model identification.

Table 2.4: CNN classification accuracy for the different sets of *Fair-60*

CNN Architectures	$\mathbb{D}_T$ accuracy (%)	$\mathbb{D}_V$ accuracy (%)	$\mathbb{D}_E$ accuracy (%)
$\mathcal{M}_{Conv4}$	97.69	97.74	94.51
$\mathcal{M}_{Conv6}$	97.82	97.53	94.67
$\mathcal{M}_{Conv8}$	97.95	97.62	94.79
$\mathcal{M}_{Conv10}$	98.01	97.81	94.93

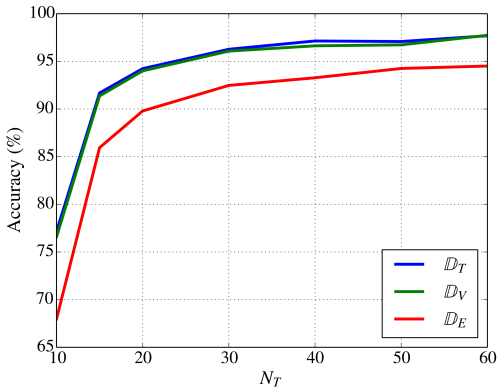
2.7.1 Camera model identification

Impact of the CNN architecture To evaluate the proposed CNN architectures, we selected a reference splitting policy showing good performance in our initial analysis. We used the *Fair-60* splitting policy and worked with a 10-fold cross validation framework (i.e., the selected splitting policy is tested 10 times on different realizations of scenes). This results in a total number of 40 trained CNN models. For evaluation, we do not to train additional SVMs, but use the CNN output as class prediction. This allows us to study the effect of the different CNN architectures on the accuracy results for a fixed splitting policy.

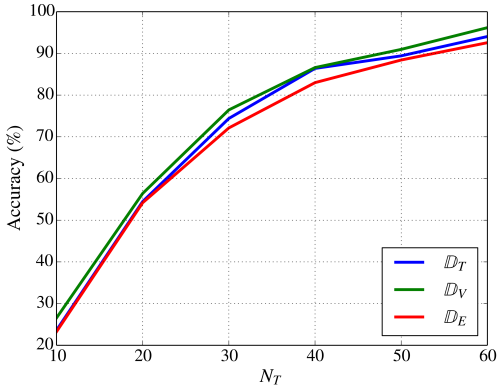
For each shot in train, validation and evaluation sets in *Fair-60*, $K = 32$ patches were extracted using the explained procedure. Training and validation patches were used to train the proposed CNNs. Specifically, the CNN architectures were trained on $\mathbb{D}_T$ patches until classification loss on $\mathbb{D}_V$ patches was minimized. Once the CNNs were trained, they were used to extract an 18 elements vector $\boldsymbol{v}_k$ for each patch $\boldsymbol{P}_k$ at the end of FullyConnected-2 layer of the CNN. Results aggregation at shot level was performed averaging element by element feature vectors $\boldsymbol{v}_k$ associated to patches $\boldsymbol{P}_k$ belonging to the same shot $\boldsymbol{I}$, so to obtain an 18 elements score vector $\boldsymbol{v}$ for the shot. Camera model associated to the maximum score was used to predict the shot's class. Shots classification accuracy was computed on $\mathbb{D}_T$, $\mathbb{D}_V$ and $\mathbb{D}_E$ as average over the 10 data realizations.

The results, presented in Table 2.4, indicate that the classification accuracy increases as we increase the CNN architecture depth: from 6 weight layers in the network $\mathcal{M}_{Conv4}$ to 12 weight layers in the network $\mathcal{M}_{Conv10}$. The camera model classification accuracy of our architecture saturates when the depth reaches 12 layers, but even deeper CNNs might be beneficial for larger datasets with a higher number of classes and training images.

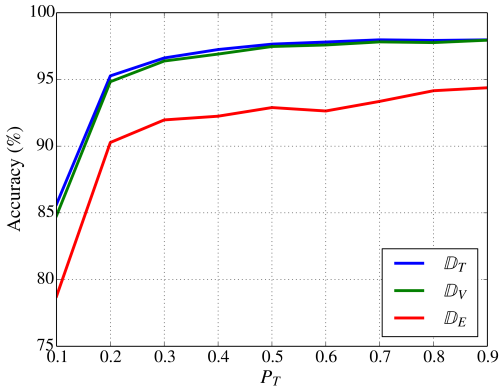
Impact of training strategy After testing different architectures, we focused on a reference CNN showing good performances and performed an extensive set of experiments over the three splitting policies described in the previous section. In particular, we selected the $\mathcal{M}_{Conv4}$ CNN detailed in Table 2.1. For evaluation, we decided not to train



(a) Fair



(b) Fair-balanced



(c) Unfair

Figure 2.8: Fair, Fair-balanced, and Unfair splitting policies results. Training (blue), validation (green) and evaluation (red) set.

additional SVMs, but to use the CNN output as class prediction. This allows us to study the effect of different training and validation split on the CNN only.

For this analysis, we fixed the number of evaluation scenes N_E to 10. For splitting policies *Fair* and *Fair-balanced* the number of training scenes N_T was varied in $\{10, 15, 20, 30, 40, 50, 60\}$ over the 73 available scenes. The remaining $73 - N_T$ scenes were used for validation. For splitting policy *Unfair* the percentage of training shots P_T was varied in $\{10, 20, 30, 40, 50, 60, 70, 80, 90\}$. The remaining shots were assigned to the validation set. This resulted in 23 splitting policies.

Figure 2.8a shows results using the *Fair* splitting policy to select train and validation datasets. Shots classification accuracy is reported as function of number of training scenes. Train and validation curves, in blue and green respectively, are almost always aligned. The red curve refers to the performance on the evaluation set. When using a small number of scenes for training (i.e., $N_t = 10$) , the small amount of data limits the CNN capabilities of learning from data. Once the number of training scenes is sufficiently large (i.e., $N_t > 15$) the results increase reaching an evaluation accuracy up to 94.5%.

Figure 2.8b shows results using the *Fair-balanced* splitting policy to select train and validation datasets. In this case the small amount of data severely limits the CNN capabilities of learning from data. In fact, in the Dresden Dataset, some camera models are represented by only a few number of shots. In the best situation (*Fair-balanced-60*), the evaluation accuracy reaches 92.6%.

Figure 2.8c shows results using the *Unfair* splitting policy to select train and validation datasets. Also in this case the small amount of training data impairs CNN learning capabilities when $P_T = 0.1$. However, as soon as the percentage of training data is increased, the evaluation accuracy reaches 94.4%.

Both the *Fair* and the *Unfair* splitting policies show a gap around 3.3% between validation and evaluation accuracies. This kind of behavior might be an indicator of some instance specific features learned during the training process.

A comparison between the *Fair* (Figure 2.8a) and the *Unfair* (Figure 2.8c) shows that there is not much gain in carefully splitting training and validation scenes. A possible motivation for this results stands in the small size of the patches used in this context. In fact, a $64 \times 64 \times 3$ patch extracted from a full resolution picture (as the ones in the Dresden Image Dataset) contains only a few details from the image, and rarely some scene specific content that might be found only in larger patches. This motivates even further the use of small patches for this learning task.

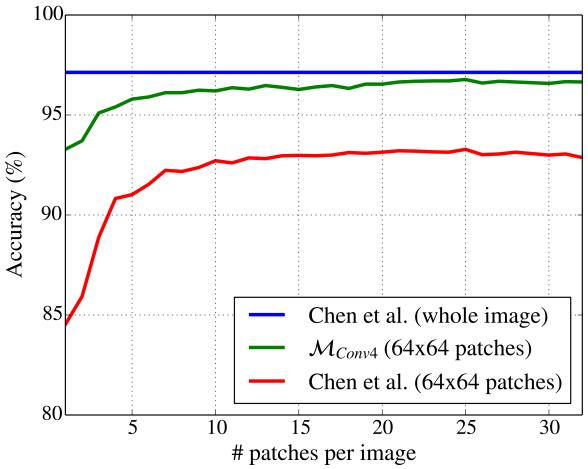


Figure 2.9: Comparison between the overall pipeline considering the CNN $\mathcal{M}_{Conv4}$ trained with *Fair-balanced-60* splitting policy and the state-of-the-art algorithm by Chen et al. [72].

Comparison with the state of the art After validating the good performance of CNNs standalone (i.e., using the FullyConnected-2 output as score for each class), we focused on the evaluation of the entire pipeline (i.e., with SVMs and majority voting) in comparison with the recently proposed state-of-the-art method by Chen et al. [72]. In particular we stopped the forward step of $\mathcal{M}_{Conv4}$ at the end of the ReLU-1 layer in order to extract from each patch P_k a feature vector $\boldsymbol{v}_k$. As dataset, we considered the *Fair-balanced-60* splitting policy.

Figure 2.9 shows how the average classification accuracy on shots from $\mathbb{D}_E$ varies while increasing the number of voting patches for each image. The proposed CNN-based approach is depicted by the green line. Benchmark result using the approach proposed by Chen et al. [72] applied on 64×64 color patches followed by majority voting is reported with red line. As the method proposed by Chen et al. is not specifically tailored to small patches, we also tested it on full resolution images without voting procedures (i.e., blue line of Figure 2.9). It is worth noting that, despite the high accuracy obtained by Chen et al., our method approaches within 1% their result by using considerably less input data (i.e., just a few patches and not the full image).

As a final remark, notice that the number of features generated at the output by the CNN for each patch is only 128, less than one tenth with respect to the $1,372$ generated by Chen et al. This confirms that we are able to characterize camera models in a space with reduced dimensionality. In principle, this enables the use of simple classifiers, which can be trained more efficiently.

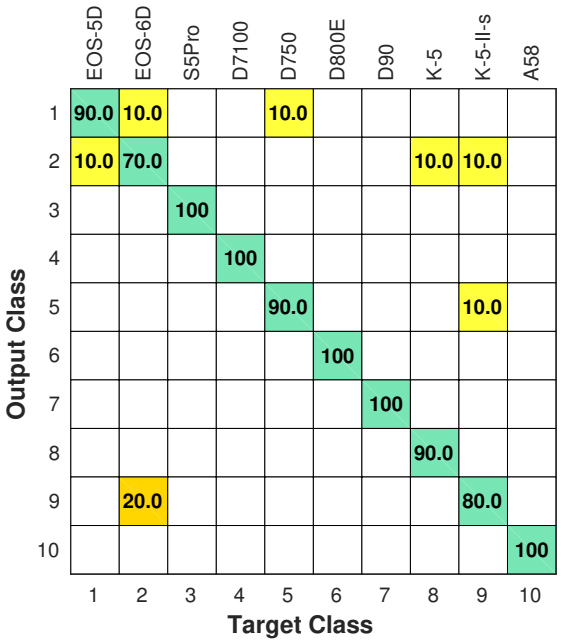


Figure 2.10: Confusion matrix for the 10-models Flickr Dataset. Results are obtained by voting with 32 patches per image. Each cell reports the percentage of images from Target Class assigned to Output Class.

CNN generalization capability One of the seemingly negative aspects of the proposed method with respect to the state of the art is that our feature extractor (i.e., the CNN) needs to be trained. Conversely, the other methods rely on manually defined feature extraction techniques. Therefore, in principle, one may think that the proposed method needs to be trained from scratch every time new camera models are considered, thus increasing the computational burden.

Our second experiment aims at disproving this preconception by showing that the feature extraction procedure learned by the CNN well generalizes to novel camera models. As a matter of fact, when new camera models are considered, only the SVMs needs to be re-trained (as it happens for classifiers used in [72].

Experimental Setup. For this experiment, we built a dataset denoted as 10-models Flickr Dataset. This was created by collecting 20 photos at full resolution from 10 different camera models (not present in the Dresden Image Dataset) from Flickr website. The dataset is split into a training set $\mathbb{F}_T$ with 10 shots per model, and an evaluation set $\mathbb{F}_E$ with the remaining 10 shots per model.

For feature extraction, we selected one $\mathcal{M}_{Conv4}$ CNN model $\mathcal{M}$ trained during the previous experiment on the Dresden Image Dataset (i.e., $\mathbb{D}_T$ and $\mathbb{D}_V$), and we adopted it on patches extracted from images in $\mathbb{F}_T$ and $\mathbb{F}_E$ (i.e., models never seen before by

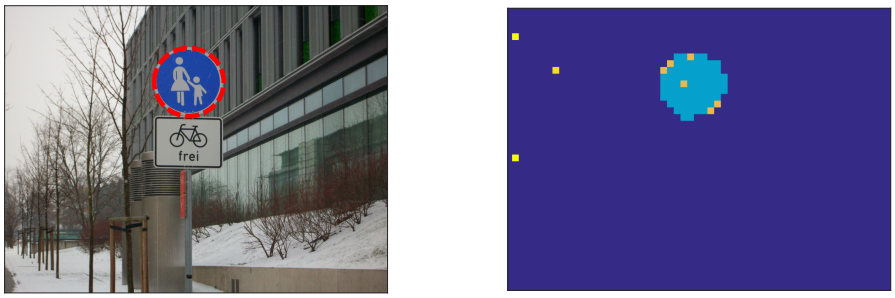


Figure 2.11: Tampering localization example. We spliced two images from different camera models (inside and outside the red circle). By classifying each block separately with $\mathcal{M}_{Conv4}$ and attributing a color to each detected camera model, it is possible to expose the splicing despite a few misclassified blocks (in yellow).

the CNN). A set of 45 linear binary SVMs was trained on feature vectors computed on patches extracted from images in $\mathbb{F}_T$. Image from $\mathbb{F}_E$ were used for evaluation.

Results. Figure 2.10 reports the confusion matrix obtained when evaluating the 10-models Flickr Dataset $\mathbb{F}_E$, using majority voting on 32 for each image. The overall accuracy reaches 93%, showing that the features extracted with the CNN model $\mathcal{M}$, trained on 18 camera models from the Dresden Image Dataset, are capable of generalizing to unknown camera models. As a matter of fact, this result suggests that the CNN learned a feature extraction procedure that is independent from the used camera models. The learned set of operations turned out to be a good procedure to expose traces characterizing different camera models. This confirms that the trained network can be used as other hand-crafted approaches [72] not requiring to be trained from scratch every time.

2.7.2 Tampering detection and localization

In this section we present all achieved experimental results regarding the detection and localization on image forgeries. To this purpose, we first present the used dataset. Finally, we report numeric results on tampering detection and tampering localization, also comparing against different state-of-the-art methods [75].

As an example of an handmade realistic example, Figure 2.11 reports a tampered image in which the background comes from a camera model, and the area marked in red comes from another model. In the example we show just the output of the $\mathcal{M}_{Conv4}$ Softmax layer, without any iteration of the iterative algorithm for tampering localization presented in Section 2.5.

Dataset The reference dataset used to validate the proposed method is the Dresden Image Database [94]. The dataset consist of more than $16k$ images from 26 different

camera models depicting a total of 83 scenes. As suggested in [29] *Nikon D70* and *Nikon D70s* are considered as the same camera model. In a first phase aimed at training the CNN, the 18 camera models with more than one device per model are taken into account and split into a training $\mathcal{D}_T$, validation $\mathcal{D}_V$, and evaluation $\mathcal{D}_E$ sets. One camera instance per model is selected for $\mathcal{D}_E$, whereas all other camera instances are in $\mathcal{D}_T$ and $\mathcal{D}_V$. Shots from same scene (e.g. outdoor, indoor, etc.) are only in one of the three sets. This replicates the splitting strategy adopted in [10], aimed at avoiding that the CNN trained on $\mathcal{D}_T$ and $\mathcal{D}_V$ over-fits on image content rather than learning camera specific features.

In a second phase two distinct tampered datasets are generated: i) the *known* dataset made from images in $\mathcal{D}_E$, and; ii) the *unknown* dataset containing images from the 8 single instance camera models never seen by the CNN. It is important in our opinion to evaluate our algorithm on both datasets to study performance differences in case known or unknown cameras are used. Indeed, working with camera models known by the CNN should be a more favorable working condition. However, it is important to notice that the algorithm is able to robustly work also with unknown cameras.

Both *known* and *unknown* datasets contain 500 pristine images and 500 tampered images generated according to the following procedure:

1. Select a random receiver image $\boldsymbol{I}_{\text{rcv}}$ from the dataset.

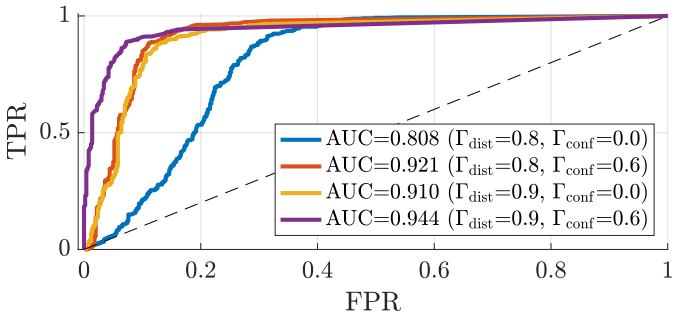
2. Generate an empty mask $\mathbf{M}$ with the same size of $\boldsymbol{I}_{\text{rcv}}$.

3. Randomly chose the number of donor alien images N_d in $[1, 2]$.

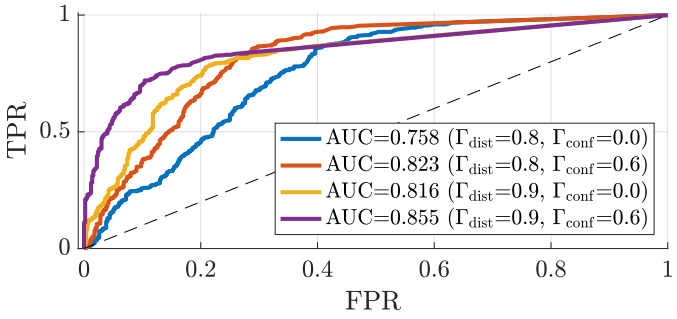
4. For each donor $d \in [1, N_d]$:
   - Select a random donor image $\boldsymbol{I}^d_{\text{dnr}}$ from the dataset, taking care it comes from a different camera model than $\boldsymbol{I}_{\text{rcv}}$.
   - Copy a rectangular random region with width and height in $[128, 1024]$ from $\boldsymbol{I}^d_{\text{dnr}}$ and paste it in a random location of $\boldsymbol{I}_{\text{rcv}}$ (not paying attention to any grid alignment).
   - Update $\mathbf{M}$ accordingly.

5. Store $\boldsymbol{I}_{\text{rcv}}$ and $\mathbf{M}$ as forged image and ground-truth mask.

Detection Each image $\boldsymbol{I}$ from the *known* and the *unknown* dataset is split into non overlapping 64×64 patches. Each patch $\mathbf{P}(i, j)$ is fed to the $\mathcal{M}_{Conv4}$ CNN and the Softmax layer output is used as a feature vector $\mathbf{f}(i, j)$ relative to $\mathbf{P}(i, j)$.



(a) *known* dataset



(b) *unknown* dataset

Figure 2.12: ROC curves of tampering detection algorithm tested on *known* (a) and *unknown* (b) datasets using different thresholds Γ_{dist} and Γ_{conf} values.

Table 2.5: Tampering detection results. Γ_{det} = best is the threshold maximizing accuracy.

Dataset	Γ_{dist}	Γ_{conf}	Γ_{det}	**ACC**	**TPR**	**TNR**
known	0.8	0.0	0	0.720	**0.994**	0.446
			best	**0.800**	0.914	**0.686**
	0.8	0.6	0	0.832	**0.980**	0.684
			best	**0.888**	0.916	**0.860**
	0.9	0.0	0	0.823	**0.966**	0.680
			best	**0.877**	0.878	**0.876**
	0.9	0.6	0	0.890	**0.942**	0.838
			best	**0.908**	0.888	**0.928**
unknown	0.8	0.0	0	0.668	**0.968**	0.368
			best	**0.731**	0.860	**0.602**
	0.8	0.6	0	0.742	**0.954**	0.530
			best	**0.784**	0.856	**0.712**
	0.9	0.0	0	0.741	**0.864**	0.618
			best	**0.785**	0.786	**0.784**
	0.9	0.6	0	0.788	**0.834**	0.742
			best	**0.810**	0.772	**0.848**

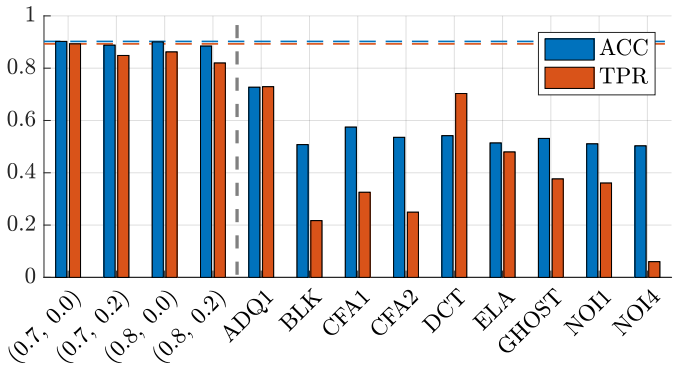
The proposed algorithm for tampering detection depends on three thresholds (i.e. Γ_{dist}, Γ_{conf} and Γ_{det}), all taking values in range $[0, 1]$. Figure 2.12 shows detection performance on both *known* and *unknown* datasets in terms of receiver operating characteristic (ROC) curves moving detection threshold Γ_{det} and fixing Γ_{dist} and Γ_{conf}. Results in terms of area under the curve (AUC) show that, on both datasets, the best detection performance are obtained for $\Gamma_{\text{dist}} = 0.9$ and $\Gamma_{\text{conf}} = 0.6$. Using Γ_{dist} close to 1 means that we only consider as forged those patches whose feature vector is far from the estimated pristine centroid. Using $\Gamma_{\text{conf}} \neq 0$ means taking into account the confidence score $\mathbf{Q}$, which actually helps in refining the detection results.

However, as the role of Γ_{det} is to decide how many forged blocks should be detected in an image in order to decide that the picture is forged, we also computed some additional results by setting $\Gamma_{\text{det}} = 0$. This case means we only detect as pristine, images for which $\hat{M}_{i,j} = 0$ for all (i, j). Table 2.5 shows results in terms of: i) true positive rate (TPR), i.e. the percentage of forged images correctly detected as such; ii) true negative rate (TNR), i.e. the percentage of pristine images correctly detected as such; iii) accuracy (ACC), i.e. the average between TPR and TNR. threshold $\Gamma_{\text{det}} =$ best, means we selected Γ_{det} values maximizing accuracy according to ROC results. Notice that, setting $\Gamma_{\text{det}} = 0$ always provide better TPR results, albeit a different threshold (i.e. the one maximizing accuracy) is more suitable to balance TPR and TNR.

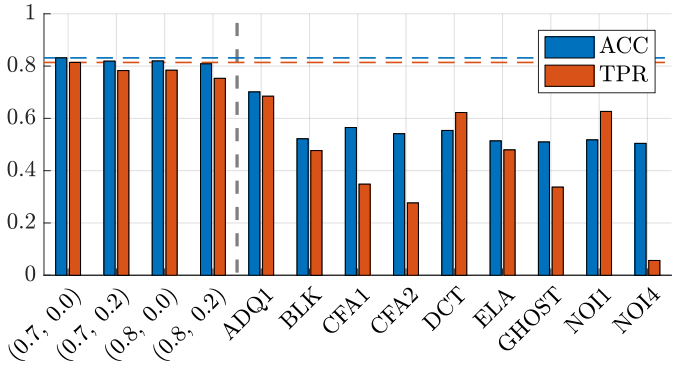
Localization As far as tampering localization is concerned, we compared the proposed algorithm on forged images against nine methods reviewed in [75], whose implementation were made available in the toolbox presented in the same paper. More specifically, we took into account all algorithms presented in Section 2.4, as they do not require any additional prior information on images under analysis (e.g. PRNUs, JPEG quantization matrix, etc.).

As evaluation metrics, we decided to rely on: i) true positive rate (TPR), i.e. the percentage of forged pixels correctly detected as such; ii) balanced accuracy (ACC), i.e. the average between TPR and the percentage of correctly detected pristine pixels. However, differently from our method, the considered state-of-the-art methods provide a soft tampering mask as output. In order to enable a fair comparison, we binarized the soft masks using the threshold that maximizes each state-of-the-art algorithm accuracy.

Figure 2.13 shows results obtained with our method for different values of thresholds Γ_{dist} and Γ_{conf}, together with state-of-the-art methods. Notice that, our method achieves best results when $\Gamma_{\text{conf}} = 0$. This means that for localization purpose, it is better to discard confidence $\mathbf{Q}$ information, which instead turned out to be paramount for



(a) *known* dataset



(b) *unknown* dataset

Figure 2.13: Accuracy and TPR on *known* (a) and *unknown* (b) datasets using our algorithm (left hand side of dashed line) with different thresholds (Γ_{dist}, Γ_{conf}) and state-of-the-art algorithms (right hand side of dashed gray line). Our best accuracy and TPR results are reported with dashed blue and orange lines.

detection purpose. Regarding comparison against state of the art, it is possible to notice that our algorithm outperforms all considered baseline solutions. This is due to the fact that considered baselines are tailored to different kinds of tampering operations. Even more interesting, the proposed method achieves promising results even on the *unknown* dataset. This means that we are able to cope with composition forgeries even when used cameras do not belong to the CNN training set. Moreover, it is possible to notice that even by slightly varying thresholds Γ_{dist} and Γ_{conf}, our results do not drop under baselines performance, showing a good degree of robustness also to the choice of sub-optimal parameters.

2.7.3 Camera model counter-forensics

In this section, we evaluate our proposed method to fool camera model identification CNN system. We compare the results of the two proposed techniques for generating

Camera Model	Training	Validation	Test
AS-One	90000	25500	12500
ES-D5100	37500	10500	5000
MK-Powershot	35000	10000	5000
MK-s860	35500	10000	5000
PAR-1233	71000	20000	10000
PAR-1476	107000	30500	15000
PAR-1477	70000	20000	9500
PAR-A015	40500	11500	5500
PAR-A075	26000	7000	3500
PAR-A106	54000	15500	7500

Table 2.6: Number of image patches per camera class for each of the different dataset splits.

the adversarial images. First, we create a reference dataset specially designed to exploit the traces left by the operations of the acquisition pipeline of different image capturing devices. Then, we train an advanced deep learning architecture to have a baseline to compare the accuracy results in the presence of adversarial images. Finally, we generate several adversarial image examples to demonstrate the performance of our proposed method.

Experimental Setup As part of DARPA's MediFor Program, PAR Government Systems collected an initial dataset of 1611 images acquired by 10 different camera models, ranging from DSLRs to phone cameras, with a mixture of indoor and outdoor flat-field scenes. We focus on a flat-field image dataset because flat-field images are more difficult to modify without inserting visual distortions due to the absence of texture content.

Throughout the rest of the section, we refer to this dataset as PRNU-PAR. Using the PRNU-PAR dataset, we create a patch dataset, composed by image patches of 32 $\times$ 32 pixels randomly extracted from the original images. Specifically, 500 patches are uniformly sampled from each original image in the PRNU-PAR dataset, which results in a patch dataset that contains 805,500 patches in total. The training, validation and test sets are created following a 70/20/10 split, while we ensure that the patches in each dataset split only contain patches from different images.

Table 2.6 shows the statistics of the patch dataset. As can be seen, due to the difference in the number of images per camera model class in the PRNU-PAR dataset, our dataset of image patches has an unequal number of patches for each of the camera models.

Figure 2.14 shows a representative example of the images that are present in the PRNU-PAR dataset next to one of their randomly extracted patches. In this case, both camera

Figure 2.14: Example of images from the training set of the patch dataset. (Top) Image from camera model PAR-A075 and one of the randomly selected patches associated with it. (Bottom) Image from camera model PAR-A106 and one of the randomly selected patches associated with it.

models PAR-A075 and PAR-A106 have been used to capture images of a cloudy sky. Other camera models such AS-One or ES-D5100 have taken images of a white screen. All the image scenes that are captured in the PRNU-PAR dataset are mostly flat and bright.

As it has been shown in the literature [28], these largely uniform images are ideal candidates to be used for the extraction of the "fingerprint" (e.g. the characteristic PRNU noise of the camera model) left in the image by the camera.

CNN architecture In order to do a fair evaluation of our counter-forensic method, we use a CNN-based camera model detector that has been trained to achieve state-of-the-art accuracy results in the patch dataset. CNN architecture designs have tended to explore deeper models. Networks which can be hundreds of layers deep are now commonplace in the literature. This design trend has been motivated by the fact that for many applications such as image classification tasks, an increase in the depth of the CNN architecture translates into higher accuracy performance if sufficient amounts of training data are available.

A first approach to design a CNN architecture may be to simply stack convolutional or fully-connected layers together. This naive strategy works initially, but gains in accuracy performance quickly diminish the deeper this kind of architecture becomes. This phenomenon is due to the way in which conventional CNNs are trained through back-

propagation. During the training phase of a CNN, gradient information must be propagated backwards through the network. This gradient information slightly diminishes as it passes through each layer of the neural network. For a CNN with a reduced number of layers, this is not a problem. For an architecture with a large number of layers, the gradient signal essentially becomes noise by the time it reaches the first layer of the network again.

The problem is to design a CNN in which the gradient information can be easily distributed to all the layers without degradation. ResNets and DenseNets are modern CNN architectures that try to address this problem.

A Residual Network [92], or ResNet is a deep CNN which tackles the problem of the vanishing gradient using a straightforward approach. It adds a direct connection at each layer of the CNN. In previous CNN models, the gradient always has to go through the activations of the layers, which modify the gradient information due to the nonlinear activation functions that are commonly used. With this direct connection, the gradient could theoretically skip over all the intermediate layers and be propagated through the network without being disturbed.

A Dense Network [56], or DenseNet generalizes the idea of a direct connection between layers. Instead of only adding a connection from the previous layer to the next, it connects every layer to every other layer. For each layer, the feature maps of all preceding layers are treated as separate inputs whereas its own feature maps are passed on as inputs to all subsequent layers. The increased number of connections ensures that there is always a direct route for the information backwards through the network. The connectivity pattern of DenseNets yields state-of-the-art accuracies on the CIFAR10 image classification dataset, which is composed by images of 32×32 pixels in size.

Motivated by the accuracy performance of DenseNet in the CIFAR10 dataset and the fact that we also work with image patches of 32×32 pixels, we select a DenseNet model with 40 layers as our CNN camera model detector. To prevent the network from growing too wide and to improve the parameter efficiency, we limit the growth rate of the network, this is, the maximum number of input feature-maps that each layer can produce, to $k = 12$. To train the CNN, we use the Adam optimizer with a learning rate of 0.0001 and a batch size of 512 images. After 5 training epochs, we reach a plateau in the accuracy in our validation set. Table 2.7 shows the single patch accuracy results for our training, validation and test splits of the patch dataset.

Adversarial image generation In order to evaluate the performance of our counter-forensic method, we test the DenseNet model trained on the patch dataset using untar-

Dataset Split	Train	Validation	Test
Accuracy (%)	99.8	98.7	97.7

Table 2.7: Single patch accuracy results for our training, validation and test splits of the patch dataset.

ϵ value	Error rate (%)	Confidence Score (%)
0.001	91.4	97.7
0.002	91.7	97.2
0.003	92.2	96.7
0.004	92.7	95.8
0.005	93.1	95.3
0.006	94.1	95.1
0.007	94.5	94.2
0.008	95.3	93.6
0.009	95.9	93.0
0.01	96.2	92.3

Table 2.8: Error rate and confidence score values of our trained DenseNet model after an untargeted attack with FGSM to the test split with different values of ϵ.

geted attacks with FGSM and targeted attacks with JSMA. To properly evaluate our method, we only perturb images from the test split which were correctly classified by our CNN in their original states.

To be clear, what we refer as the average confidence score is the average value of the probability that is associated with the candidate camera model label for each of the image patches in the test split. The probability for each candidate camera model label corresponds with the highest probability value assigned by the softmax layer of our trained DenseNet model.

For untargeted attacks with FGSM, we report in Table 2.8 the error rate and the average confidence score on the test split of the patch dataset for different values of ϵ which have been shown to generate high misclassified adversarial images while not producing appreciable visual changes. We find that using $\epsilon = 0.005$ offers the best compromise between error rate and visual changes in the image, causing the trained DenseNet model detector to have a error rate of 93.1% with an average confidence of 95.3% on the patch test split. It should be noted that as we increase the value of ϵ, the manipulations become more visually apparent.

Figure 2.15 shows an example of the adversarial images that our proposed method can generate when we use FGSM. The modifications done to the images by FGSM are

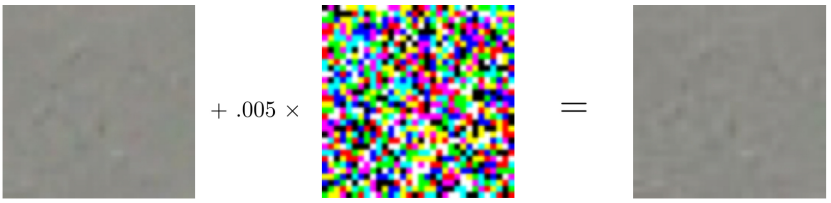


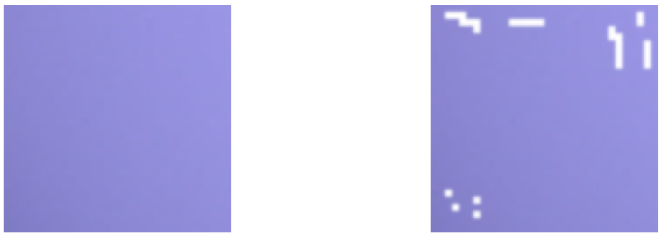
Figure 2.15: An example of untargeted fast adversarial image generation using FGSM applied to our trained DenseNet model on the patch dataset. By adding an imperceptibly small vector whose elements are equal to the sign of the elements of the gradient of the cost function with respect to the input, we can change DenseNet's classification of the image patch.

Target Camera Model	**Error rate (%)**	**Confidence Score (%)**
AS-One	99.5	87.7
ES-D5100	99.3	88.6
MK-Powershot	99.3	88.4
MK-s860	99.7	88.5
PAR-1233	99.7	87.9
PAR-1476	99.4	88.1
PAR-1477	99.5	88.2
PAR-A015	99.6	88.4
PAR-A075	99.3	87.8
PAR-A106	99.2	87.9

Table 2.9: Error rates and confidence scores of our trained DenseNet model for each possible target camera model after applying a targeted attack with JSMA to the test split.

performed on 32-bit floating point values, which are used for the input of the DenseNet model. The gradient computed for Figure 2.15 uses 8-bit signed integers. The image representing the sign of the gradient is converted from 8-bit signed integers to 8-bit unsigned integers. To increase the range of each color channel, we represent the -1s values as 0 and the 1s as 255. For the possible 0's, we have treated them as positive values (they are represented by 255).

For targeted attacks with JSMA, we report in Table 2.9 the error rate and the average confidence score for each possible camera model target class. Figure 2.16 shows an example of the images that JSMA allows us to generate when we perform a targeted attack. In this case, an image patch captured by camera ES-D5100 that is correctly classified when is analyzed by our trained DenseNet model is manipulated to be misclassified as an image



Detected camera model: ES-D5100
98.1% confidence

Detected camera model: PAR-1233
89.6% confidence

Figure 2.16: An example of targeted adversarial image generation using JSMA applied to our trained DenseNet model on the patch dataset. (Left) Original image patch correctly classified as ES-D5100. (Right) Altered image patch with target camera model PAR-1233

patch that had been generated by camera model PAR-1233. It is important to appreciate that although JSMA allows us to generate image patches that get misclassified into a specific camera model with high error rates and confidence scores, the modifications that it applies to the images can usually be spotted through visual inspection. This effect is due to the fact that JSMA crafts the adversarial images by flipping pixels to their minimum or maximum values. Because our patch dataset is composed of image patches with mostly flat scene content, the effect can be clearly observed, for example, in the upper corners of the manipulated image patch in Figure 2.16.

2.8 Conclusions

In this chapter we presented the possibility of using CNNs to solve camera model identification problems. The proposed algorithm has been validated paying particular attention to the used evaluation protocol in order to avoid overfitted or biased results. We investigated the effect of training a CNN on different data splits in order to highlight the dependency between accuracy, training set size, and training-testing splitting policy. This study shows that it is possible to achieve high camera model attribution accuracy (i.e., around 96%) even with fairly small network architectures (i.e., four convolutional layers), provided that a minimum amount of training images are available. Indeed, the use of larger configurations determines a negligible accuracy increment, at least on the selected dataset of 18 camera models. Comparison against state-of-the-art method showed promising results when small image patches are considered. Accuracy achieved in the literature exploiting full size images can be obtained with our method with a considerable reduced amount of information (i.e., a few patches).

Despite being a method based on deep learning, results show that the CNN can be

trained only once to learn a feature extraction procedure that can be used afterwards also on images from camera models never seen by the CNN. Moreover, the ability of associating small image patches to camera models, paves the way to the development of image splicing localization methods.

With this regard, the method proposed in Section 2.5 exploits a the CNN model presented in Section 2.4 to extract features capturing camera model traces from image patches. Features are clustered into two groups (i.e. pristine or forged) using an iterative algorithm, and information about patches content and locations is further used to refine forgery localization estimation.

Evaluation carried out on a dataset of 2000 images obtained from 26 camera models shows that the proposed algorithm is able to detect forged images with accuracy of 0.91 if camera models involved in forgeries have been used during CNN training. If camera models involved in forgeries have never been used for training, detection accuracy still remains as high as 0.81. Results on tampering localization shows that it is possible to detect forged regions with accuracy about 0.90 and 0.82 depending on the knowledge (or not) of the used camera models at training stage. This makes the proposed method more accurate than alternative baseline solutions selected from [75].

Finally, we proposed a counter-forensic method to subtly alter images to change their estimated camera model when they are analyzed by a CNN-based camera model detector. We tested our method on a reference dataset with images from multiple cameras that show highly similar indoor and outdoor scenes. The results demonstrate that we can generate imperceptibly altered adversarial images that are misclassified with high confidence by the CNN, opening the way to a new class of counter-forensics attacks. The challenge in designing and training a CNN architecture for camera model identification that is accurate on a high number of camera models and robust against adversarial attacks is still open.

3 Methods based on sensor fingerprints

3.1 Introduction

Over the last decade, ownership attribution and origin verification of digital content has become of capital importance, due to the widespread diffusion of digital devices capable of acquiring images, videos and audio tracks, as well as sharing them over the Internet. Active forensic methods based on image watermarking or bit-stream embedded metadata are not always applicable for owner identification purposes, or may be easy to fool. Watermarks, for instance, must be inserted at image inception time to let the content be recognizable even after common transformations such as rotation, scaling, or compression [96]. Conversely, metadata manipulation is at everyone's hand, and with a few clicks it is possible to anonymize an image removing all its EXIF properties [97]. To face ownership attribution in a completely blind fashion, researchers have developed a set of techniques tailored to extract traces left on the image by processing components such as lens aberrations [57, 98], color filter array (CFA) demosaicing artifacts [58, 99], JPEG compression traces [100], or combinations of these [27, 101].

The most widespread technique for camera device identification is based on the Photo Response Non-Uniformity (PRNU) [13]. This is a time invariant weak multiplicative signal introduced on every picture taken with a CCD/CMOS imaging device. PRNU traces are mainly due to sensor's silicon imperfections and can be exploited as a unique fingerprint for each camera sensor. Silicon imperfections occurring at sensor manufacturing process cause each pixel to have a slightly different area, thus the amount of light energy captured in a fixed time slot (i.e. exposure time) varies pixel-wise even under a perfectly uniform light field. Such a fingerprint is embedded in every shot taken with a specific device, being it a professional Digital Single-Lens Reflex (DSLR) camera or a cheap smartphone. PRNU knowledge allows to determine whether two images have been captured by the same device, link a picture to the specific camera that took it, and even detect forgeries [102–104].

The PRNU can be extracted, or at least estimated, by having access to a set of images captured by the same camera. As a matter of fact, the PRNU fingerprint extraction

process from natural images has been deeply analyzed [105–110] and the effect of image content residuals after noise extraction has been faced with signal enhancement techniques [111]. Moreover, the resilience of the extracted PRNU traces to cropping, scaling and JPEG compression [112, 113] gave way to a massive usage of PRNU fingerprints for camera device identification [114]. Furthermore, clustering based approaches [115, 116] have shown the possibility of separating pictures coming from several camera devices based on the analysis of PRNU traces.

PRNU compression

One drawback of large scale approaches is the need to store in a central database, or transmit over bandwidth-limited channels, a huge amount of data. Indeed, PRNU fingerprints need to be extracted at higher resolutions, up to the size of the imaging sensor, to achieve better matching and detection performance and avoid false-alarms. In a large-scale retrieval setup the need of storing several thousands of reference fingerprints at full resolution poses issues regarding the amount of storage space. A second issue arises in terms of computational complexity, when a query fingerprint needs to be matched against many device fingerprints stored in a central camera fingerprints database. In a mobile authentication scenario [117], when a user wants to authenticate its device by sending a residual extracted from a picture to a centralized database for verification, restricting the amount of data being sent over the network is mandatory to reduce response times and improve user experience.

An overview of the state-of-the art techniques on PRNU compression is provided in Section 3.3.1, while in Section 3.4 we address the problem of PRNU compression by proposing:

1. a projection design methodology that takes into account interpolation effects on the PRNU (Section 3.4.2)

2. a pipeline for PRNU compression tailored to JPEG images based on state-of-the-art Random Projections [15] (Section 3.4.3)

As for the former, a Signal-to-Noise-Ratio (SNR) maximization problem for the alternative hypothesis of a fingerprint cross-correlation test is solved, taking into account the interpolation effect due to the post-acquisition operations a digital image undergoes, e.g. demosaicing, JPEG compression. We first provide a theoretical framework to establish near-optimal conditions for the projection matrix, then we provide a design methodology for such a matrix. The approach not only yields better detection performance as

compared to the state of the art, but is much cheaper in computational terms, which is a critical indicator when huge databases need to be searched.

As for the latter, we leverage on two founding concepts for PRNU compression tailored to JPEG images:

- As images are often JPEG compressed, high-frequency PRNU components may be corrupted, thus making low-frequencies overall more informative.
- Dead-zone quantization provides much more flexibility than binary one, especially when paired with entropy coders.

Exploiting the first idea, we show that it is possible to pre-process PRNU traces with a decimation operation up to a certain ratio before reducing vector space dimensionality with Random Projections, still retaining important camera device information. Even though PRNU is robust to JPEG compression in terms of device identification or verification [13], the strong quantization introduced by JPEG compression at high frequency components [62, 118] motivates the introduction of a low-pass filtering via decimation as first step of the pipeline, in order to preserve only those frequency components that are carrying significant information about the original PRNU signal.

The second idea basically compromises between fingerprint binarization [119] (i.e., binary quantization) and fingerprint digest [50] (i.e., only coding prominent peaks). Thanks to the proposed dead-zone quantization process, it is possible to further compress fingerprints exploiting entropy coders that did not prove useful after PRNU binary quantization.

Image anonymization

When privacy is a concern, being able to link a picture to its owner is clearly undesirable. As an example, photoreporters carrying out legit investigations may prefer to anonymize their shots in order to avoid being threatened. For this reason, counter-forensic methods that enable deleting or reducing PRNU traces from images have been proposed in the literature. An overview is offered to the reader in Section 3.3.2.

In Section 3.5 we investigate different solutions for PRNU anonymization:

1. two parallel and fast inpainting techniques as methods for image anonymization that do not require the knowledge of the camera fingerprint (Section 3.5.2)
2. an auto-encoder inspired fully-convolutional neural network for image anonymization, that instead requires the knowledge of the camera fingerprint (Section 3.5.3)

Inpainting methods Inpainting is a well-known topic, which refers to the application of simple or sophisticated algorithms for reconstructing lost or corrupted portions of an image (e.g., by solving partial differential equations, with the application of sparse domain transformations or the regularization of inverse problems, etc.). The rationale behind our approach is that, by deleting and reconstructing each pixel from its neighbors, PRNU effect can be strongly attenuated. The proposed method mainly works in two steps: (i) each image pixel is substituted by its inpainted value, in order to corrupt the PRNU; (ii) pixels around edges are replaced by denoised versions of the original ones in order to mask possible visual artifacts around sharp discontinuities.

Even though the literature is wide and provides many advanced solutions, we investigate the use of simple yet effective inpainting schemes that keep computational complexity at bay. Specifically, we only consider solutions that reconstruct pixels by solving inverse regularized problems.

In particular, it is important to point out that a regularization which perfectly reconstructs the original image would lead to a high visual quality result, but would be totally useless for what concerns the anonymization task, as the traces of PRNU would remain almost unchanged. Therefore, in order to achieve our goal, the algorithm proposes a trade off between quality and anonymization of the outputs.

Auto-encoder method We explore the possibilities offered by CNNs in terms of camera device anonymization based on the knowledge of the reference PRNU. An image-wise anonymization loop is built upon a CNN-based noise extractor. An auto-encoder inspired fully-convolutional neural network is trained as anonymization function via back-propagation, exploiting the possibilities offered by a recently introduced CNN-based denoising method [120].

The proposed use of a CNN is different from the typical one. Instead of training a CNN on many images to learn a generalizable method, we "overfit" the proposed CNN on each single image to be anonymized. In other words, we consider the CNN as a parametric operator. We build a loss function to be minimized in order to estimate the CNN parameters. The CNN training is seen as an iterative way of minimizing the CNN loss for each given image.

3.2 Background

In this section we introduce the necessary background about PRNU fingerprint estimation and matching, and finally we present a short overview about JPEG compression and how

it affects PRNU fingerprints and residuals.

3.2.1 PRNU estimation

Photo Response Non-Uniformity (PRNU) is a multiplicative noise pattern mostly related to different sizes of imaging sensor cells. It is a weak signal caused by minute imperfections occurring during the manufacturing process of the sensor. Despite its weakness, when a sufficiently large number of image samples is available it is possible to estimate and use it as a robust fingerprint for a specific camera sensor [105].

$$\boldsymbol{y} = g^{\gamma}\left[(\mathbf{1} + \boldsymbol{k}) \circ \boldsymbol{f} + \boldsymbol{z}\right]^{\gamma} + \boldsymbol{n}_q \tag{3.1}$$

where $\boldsymbol{y}$ is the one-dimensional representation of the acquired image $\boldsymbol{Y}$, $\boldsymbol{f}$ is light field at the sensor, g is the color channel gain and γ is the gamma correction factor [103]. $\boldsymbol{k}$, the column-wise unwrapping of $\boldsymbol{K}$, is a zero-mean noise-like signal identified as the PRNU fingerprint, $\boldsymbol{z}$ is a combination of remaining noise sources (dark currents, read-out noise, shot noise), and $\boldsymbol{n}_q$ is quantization noise. The imaging model can be further simplified as

$$\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{x} \circ \boldsymbol{k}_0 + \boldsymbol{n} \tag{3.2}$$

where $\boldsymbol{x} = (g\boldsymbol{f})^{\gamma}$ is a noise-free version of $\boldsymbol{y}$, $\boldsymbol{k}_0 = \gamma\boldsymbol{k}$ and $\boldsymbol{n}$ condensates all the independent random noise components residuals. Here we consider, without loss of generality, the wavelet decomposition based denoising algorithm proposed in [121] and adopted in [102] to get $\boldsymbol{x}$. When a single query image $\boldsymbol{y}_q$ is available, we define its residual as

$$\boldsymbol{w}_q = \boldsymbol{y}_q - \boldsymbol{x}_q \ . \tag{3.3}$$

When a set of N_c images from the same device is available, we define the residual for each picture $\boldsymbol{y}_c,\ c \in [1, N_c]$ as $\boldsymbol{w}_c = \boldsymbol{y}_c - \boldsymbol{x}_c$ and we obtain the estimated PRNU fingerprint $\hat{\boldsymbol{k}}$ for the device as

$$\hat{\boldsymbol{k}} = \frac{\sum_{c=1}^{N_c} \boldsymbol{w}_c \circ \boldsymbol{y}_c}{\sum_{c=1}^{N_c} \boldsymbol{y}_c^2} \tag{3.4}$$

Wiener adaptive filtering in the Discrete Fourier Transform domain and mean subtraction are finally applied to $\hat{\boldsymbol{k}}$ in order to remove model or compression specific artifacts.

When a query image under investigation $\boldsymbol{y}_q$ and a camera device c, characterized by a PRNU fingerprint $\boldsymbol{k}_c$ are made available, a two-channel hypothesis testing problem is faced in order to determine whether query $\boldsymbol{y}_q$ has been shot by the device characterized

by $\boldsymbol{k}_c$:

$$\begin{aligned} H_0 :&\ \boldsymbol{y}_q \text{ was not taken with camera } c; \\ &\ \text{thus it does not contain } \boldsymbol{k}_c \\ H_1 :&\ \boldsymbol{y}_q \text{ was taken with camera } c; \\ &\ \text{thus it contains } \boldsymbol{k}_c \end{aligned}$$

Detection of such matching is performed via cross-correlation test, defined as

$$r \doteq \langle \hat{\boldsymbol{k}}_c, \boldsymbol{w}_q \rangle = \sum_{i=1}^{N} \hat{k}_{c,i} \cdot w_{q,i} \tag{3.5}$$

When $r > \tau$ then H_1 hypothesis is verified and query image is verified to come from c. τ is a threshold on the cross-correlation test properly set in order to bound the false-alarm probability under a maximum desired value.

3.2.2 JPEG compression

JPEG compression is the most widespread standard for saving natural pictures in a digitized way. All camera models and smartphones, both professional and cheap ones, provide a way to save on non-volatile storage the acquired images in JPEG format.

At first, a color space transformation from the RGB color space to the YCbCr color space is applied to the original image, to get a luma component (Y) and two chroma components (Cb, Cr). Luma and chroma matrices are split in 8×8-pixel non-overlapping blocks. Every block is transformed with 2D Discrete Cosine Transform, rearranging the 64 resulting coefficients in a 8×8 matrix where the top-left element contains the DC component and the bottom-right element contains the highest – both vertically and horizontally – frequency component coefficient.

Each block is then quantized by dividing each frequency component by a specific quantization step, then rounding the result to the nearest integer. Quantization coefficients are stored in two quantization matrices, one for luma component and one for chroma components. The aforementioned coefficients quantization carries two effects: i) a reduction of inter-coefficient entropy, exploited by zig-zag run-length Huffman coding to compress each block thus reducing storage space; ii) a frequency-dependent filtering, that greatly reduces high-frequency components while preserving low-frequency ones.

The final effect of JPEG compression on an image is a block-wise low-pass filtering. This also affects the image embedded PRNU, which loses its white-shaped spectrum in

favor of a low-pass version. This consideration stands behind the choice of pre-processing with a low-pass filter the estimated PRNU fingerprints and image residuals, in order to reduce the amount of data being processed and transmitted, while preserving the surviving spectral components of the PRNU.

3.3 State of the art

In this section we present the most relevant techniques at the state of the art in PRNU compression, Section 3.3.1, and PRNU-based antiforensics techniques, Section 3.3.2. Some of the presented techniques are used as baseline to compare the results of the proposed methods in the following sections.

3.3.1 PRNU compression

Being the problem of PRNU dimensionality reduction necessary in real-world applications, several PRNU compression techniques have been proposed in the literature. In the following, we present an overview of the most relevant dimensionality reduction systems presented in the literature. The illustrated compression strategies are described in terms of camera fingerprint ($\hat{\boldsymbol{K}}_c$) but the same process holds also for query residuals ($\mathbf{W}_q$).

Trimming and cropping Fingerprint trimming [50] is the most trivial way of compression. Considering $\hat{\mathbf{k}}_c$ as the column-wise unwrapping of $\hat{\boldsymbol{K}}_c$, trimming is performed by preserving only the first P samples from $\hat{\mathbf{k}}_c$.

Similarly, fingerprint cropping results when preserving only the central portion of a fingerprint $\hat{\boldsymbol{K}}_c$ and then performing the unwrapping.

Digest Fingerprint digest [50] is one of the most effective PRNU compression techniques. The base idea is that most prominent peaks of the extracted PRNU $\hat{\boldsymbol{K}}_c$ are more relevant when using a cross-correlation test for device attribution. Therefore, the digest is built by retaining the position and value of the P highest energy pixels from $\hat{\boldsymbol{K}}_c$, creating a pair of vectors of length P, holding respectively peak values and positions.

While this method turns out to be very effective in terms of compression ratio, it requires the knowledge of a rather good estimate of $\hat{\boldsymbol{K}}_c$ to preserve those pixels that are really characterizing a specific sensor fingerprint. This is made possible only when $\hat{\boldsymbol{K}}_c$ is estimated from many images, as only with a good estimate of the PRNU the peaks selection process is robust enough to allow compression while retaining high detection performance.

In a query compression scenario, as in a joint compression scenario, where query residual compression is limited to the knowledge of a single image, fingerprint digesting is not a viable option.

Binarization Fingerprint binarization [119] is an effective way to greatly reduce the fingerprint bitrate even after trimming, cropping or projection. Binarization is defined as an element-wise operation transforming a real number x into its binarized version x_b as

$$x_b = \begin{cases} +1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases} \tag{3.6}$$

An additional benefit from binarization is the reduced computational complexity when performing a cross-correlation test, as shown by Bayram et al. [119].

Gaussian Random Projections Introduced as PRNU compression method by Valsesia et al. [15], Random Projections (RP) with Gaussian sensing matrix have proven to be an effective way of compressing camera fingerprints and query residuals. The idea of RP is to project the one dimensional unwrapping $\hat{\mathbf{k}}_c$ of fingerprint $\hat{\boldsymbol{K}}_c$ from a vector space of dimension $L = h \cdot w$ to a subspace of dimension P.

Formally, a sensing (projection) matrix $\boldsymbol{\Phi}$ with size $L \times P$ is generated with samples being extracted from a i.i.d. zero-mean Gaussian distribution. The resulting projection $\boldsymbol{r}_c$ is thus a matrix product between a sensing matrix $\boldsymbol{\Phi}$ and a vector $\hat{\mathbf{k}}_c$

$$\boldsymbol{r}_c = \boldsymbol{\Phi} \odot \hat{\mathbf{k}}_c \tag{3.7}$$

In order to speed-up computation and save memory, a simplified way of building the projection matrix is to randomly generate as i.i.d. zero-mean Gaussian a single column $\boldsymbol{\phi}$ of $\boldsymbol{\Phi}$, then generate all other columns by means of circular shifting. In this way the projection can be turned into an element-wise product in the Fourier Transform domain as

$$\boldsymbol{r}_c = \mathrm{IFFT}\left(\mathrm{FFT}(\hat{\mathbf{k}}_c) \circ \mathrm{FFT}(\boldsymbol{\phi})\right) \tag{3.8}$$

The process ends by preserving only the first P elements of $\boldsymbol{r}_c$. The binary version of $\boldsymbol{r}_c$ is denoted as r_c^b.

3.3.2 Antiforensics

Given an image $\boldsymbol{Y}$ shot with a camera whose PRNU fingerprint is $\boldsymbol{K}_c$, the cross-correlation test between the PRNU residual $\boldsymbol{W}$ extracted from $\boldsymbol{Y}$ and $\boldsymbol{K}_c$ is expected to be greater than a threshold τ. Formally, $\langle \boldsymbol{K}_c, \boldsymbol{W} \rangle > \tau$.

PRNU-based antiforensics techniques aim at modifying $\boldsymbol{Y}$ to generate an anonymized version $\tilde{\boldsymbol{Y}}$ so that

- The PRNU residual $\tilde{\boldsymbol{W}}$ extracted from $\tilde{\boldsymbol{Y}}$ fails the cross-correlation test with $\boldsymbol{K}_c$. Formally, $\langle \boldsymbol{K}_c, \tilde{\boldsymbol{W}} \rangle < \tau$.
- The perceptual quality of the anonymized image is as similar as possible to the original one. One of the measures used to evaluate the perceptual quality is the Peak Signal-to-Noise Ratio (PSNR) between $\tilde{\boldsymbol{Y}}$ and $\boldsymbol{Y}$, whose value should be as high as possible.

Among the developed techniques, some require the knowledge of the PRNU fingerprint to be deleted, while others are PRNU independent.

PRNU-aware techniques In [122] the authors propose an adaptive fingerprint removal technique based on what suggested in [105,123]. The base idea is to remove the estimated camera fingerprint $\boldsymbol{K}_c$ from the original image $\boldsymbol{Y}$ to obtain an anonymized image $\tilde{\boldsymbol{Y}}$ as

$$\tilde{\boldsymbol{Y}} = \boldsymbol{Y} - \beta \boldsymbol{K}_c \tag{3.9}$$

where β is a magnitude adjustment estimated by solving

$$\langle \tilde{\boldsymbol{Y}}, \boldsymbol{K}_c \rangle \approx 0 \tag{3.10}$$

By substituting the cross-correlation in Equation (3.10) with the Peak to Correlation Energy (PCE), as from [124], and changing the type of PRNU extractor used to estimate the image residual, the authors of [122] improve the PSNR between $\boldsymbol{Y}$ and $\tilde{\boldsymbol{Y}}$ while obtaining better anonymization performance.

One of the challenges in adaptive fingerprint removal is that $\boldsymbol{K}_c$ estimate, $\hat{\boldsymbol{K}}_c$, can be obtained from different images and using several denoising filters. Based on this consideration, the authors of [125] propose an iterative anonymization algorithm based on two different estimates of $\boldsymbol{K}_c$. One estimate, $\boldsymbol{K}_c^{(1)}$, is used to attenuate the PRNU traces, as in Equation (3.9), the other, $\boldsymbol{K}_c^{(2)}$, is used to modify the value of β. The algorithm iterates until the stopping condition $\left| \langle \boldsymbol{K}_c^{(2)}, \boldsymbol{W} \rangle \right| < \tau$ is met or the maximum number of iterations is reached.

PRNU-independent techniques The authors of [113] have studied the effect that multiple denoising steps and multiple lossy compression-decompressions have on PRNU-based image source identification. They noticed that, when repeated multiple times, both sets of operations behave as anonymizers. However, the cost in terms of image quality loss is generally too high.

Given the nature of the cross-correlation test, Equation (3.5), image resizing with tailored seam-carving techniques is introduced in [126] to de-synchronize the positions of the pixels between the anonymized image and the reference PRNU fingerprint. By forcing seams over specific pixel locations, the anonymized image is perceptually very similar to the original one, but the largest synchronized (i.e. unmodified) block in the anonymized image is not large enough to have sufficient statistics to correctly match in a block-wise cross-correlation test with the reference PRNU. The main drawback of such a technique is that the number of seams to be removed in order to get good anonymization performances is generally too high, thus compromising the anonymized image quality.

The most effective PRNU-independent anonymization technique for sensor-based source identification is presented in [20]. Based on PatchMatch [127], as the seam-carving approach it is based on the idea of de-synchronizing the PRNU residual extracted from an anonymized image. Each patch of the original image is substituted by a weighted average of similar patches coming from the same image. As the most similar patch within the same image is always the patch itself, the PatchMatch algorithm is modified so to enforce a minimum mean squared error between the source patches and destination patch.

3.4 Compression

In this section we formalize the problem of PRNU compression, Section 3.4.1, and we propose two different approaches to tackle it.

The first approach, Section 3.4.2, is based on the design of projection matrices that, under hypothesis H_1, maximize the Signal-to-Noise Ratio between the expected value and the standard deviation of the cross-correlation test. The main goal of this approach is complexity reduction by designing computationally efficient projection matrices.

The second approach, Section 3.4.3, is tailored to fingerprints and residuals extracted from JPEG compressed images, and aims at designing a pipeline for PRNU compression that empowers the state-of-the-art Random Projections. The goal is to obtain a better compression rate while preserving unaltered detection performance.

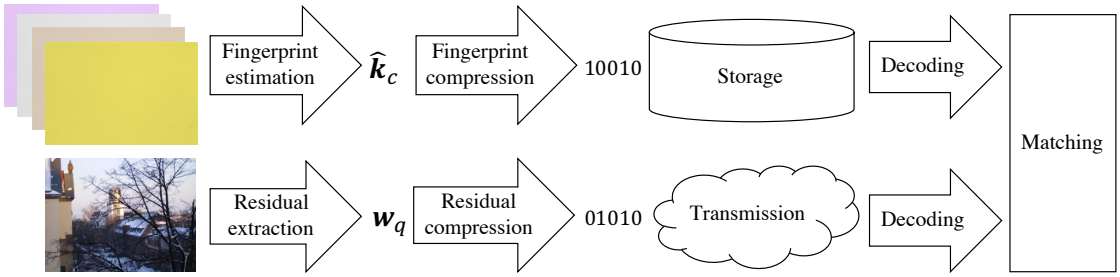


Figure 3.1: Overall database and query pipeline. (Top) A set of flatfield images is used to estimate a camera fingerprint $\hat{\boldsymbol{k}}_c$, that is compressed and stored into a database. (Bottom) Residual $\boldsymbol{w}_q$ is extracted from a single query image, then compressed and sent to a central location for matching purpose.

3.4.1 Problem formulation

When it comes to storing a huge amount of fingerprints or there is the need of sending them over a band-limited communication channel, an effective PRNU compression method becomes mandatory.

We are interested in two main applicative scenarios that can summarize several real-world applications (see Figure 3.1). Both scenarios include two players: i) a central database that stores camera fingerprints, each extracted from several images; ii) a number of query devices whose fingerprints need to be sent to a central server for matching purposes. The two scenarios are described in the following:

Query compression scenario The goal is to reduce as much as possible the amount of memory used to represent query residual information. This means minimizing the bitrate required to send the compressed residual from a remote device to a central server. Equivalently, this can be interpreted as minimizing the file size in case of residual storage applications.

Joint database-query compression scenario The first goal is to restrict as much as possible both the bitrate required to send a query residual to a central server and at the same time limit the storage space required to store camera fingerprints. The second goal is to reduce the computational complexity required to match each camera fingerprint with a given query residual.

One of the main issues in compressing a PRNU fingerprint comes from the observation that it is an inherently broadband white noise-like signal, well modeled as a sequence of i.i.d. samples drawn from a zero mean Gaussian distribution. This results in a signal with little or no redundancies to be exploited for lossless compression. Lossy compression is then the only way to reduce fingerprint's rate. In particular, we focus on compressing

PRNU fingerprints and residuals when no geometrical transformations are applied to the original images.

From a formal point of view, the problem of PRNU compression faced in this work can be defined as follows. Let $\mathcal{C}$ be a collection of camera fingerprints, where each fingerprint is estimated from several flatfield images according to Equation (3.4). Let $\mathcal{Q}$ be a collection of query residuals, where each residual is extracted from a single query image. Let $\hat{\boldsymbol{k}}_c$ and $\boldsymbol{w}_q$ be an estimated fingerprint and a noise residual extracted respectively from $\mathcal{C}$ and $\mathcal{Q}$. The main goal is to generate reduced rate representations of $\hat{\boldsymbol{k}}_c$ and $\boldsymbol{w}_q$ such that

- the performances in terms of Receiver-Operating-Characteristic of the compressed and the uncompressed case are similar.
- storage space – or transmission rate – requirements of the compressed fingerprints and residuals are minimized.

In Section 3.4.2 we present a projection matrix design methodology tailored to complexity reduction by designing computationally efficient projection matrices.

In Section 3.4.3 we design a compression pipeline with the goal of obtaining a better compression rate, while preserving unaltered detection performance.

3.4.2 Projection matrix design

Given a device whose estimated PRNU fingerprint is $\hat{\boldsymbol{k}}_c \in \mathbb{R}^N$ and a query image $\boldsymbol{y}_q \in \mathbb{R}^N$, our goal is to design a projection matrix $\boldsymbol{\Phi}$ of size $L \times N, L < N$ that under statistic

$$r \doteq \langle \boldsymbol{\Phi}\hat{\boldsymbol{k}}_c, \boldsymbol{\Phi}\boldsymbol{w}_c \rangle \tag{3.11}$$

enables dimensionality reduction of both $\hat{\boldsymbol{k}}_c$ and $\boldsymbol{w}_c$ while maximizing detection probability for H_1 hypothesis and minimizing false alarm probability for H_0 hypothesis.

We assume the existence of a zero-mean vector $\boldsymbol{k}_0$ with N i.i.d. components $\mathcal{N}(0, \sigma_k^2)$ describing the PRNU of a sensing device.

The estimated device PRNU fingerprint $\hat{\boldsymbol{k}} \in \mathbb{R}^N$, extracted from a set $\mathcal{C}$ of images taken with the same camera device as from Equation (3.4), is modeled as

$$\hat{\boldsymbol{k}} = \boldsymbol{H}_c\boldsymbol{k}_0 + \boldsymbol{n}_e \tag{3.12}$$

where $\boldsymbol{n}_e$ is the extraction noise which we assume zero-mean i.i.d., i.e. $\mathrm{E}\{\boldsymbol{n}_e\boldsymbol{n}_e^T\} = \sigma_e^2\boldsymbol{I}$. Matrix $\boldsymbol{H}_c \in \mathbb{R}^{N\times N}$ accounts for demosaicing effects in the spatial domain and possibly

the equivalent filtering performed in the DCT domain when images are compressed. All other effects due to compression are included in $\boldsymbol{n}_e$.

Given a query vector $\boldsymbol{y}$ from an image in the query images set $\mathcal{Q}$, and its denoised version $\boldsymbol{x}$, we model $\boldsymbol{x} = \mu_x \mathbf{1} + \tilde{\boldsymbol{x}}$, where $\tilde{\boldsymbol{x}}$ is assumed to be stationary with autocorrelation $r_{\tilde{x}}[i-j] \doteq E[\tilde{x}_i \tilde{x}_j]$.

The noise residual $\boldsymbol{y} - \boldsymbol{x}$ is modeled as $\boldsymbol{H}_q(\boldsymbol{k}_0 \circ \boldsymbol{x}) + \boldsymbol{n}_t$, where $\boldsymbol{n}_t$ is zero-mean i.i.d. noise, i.e. $\mathrm{E}\{\boldsymbol{n}_t \boldsymbol{n}_t^T\} = \sigma_t^2 \boldsymbol{I}$ and $\boldsymbol{H}_q$ plays a similar role to $\boldsymbol{H}_c$. In general $\boldsymbol{H}_c \neq \boldsymbol{H}_q$ because we assume that images from $\mathcal{C}$ and $\mathcal{Q}$ may be compressed differently. Also notice that more complicated noise models can be easily accommodated in our discussion, but we keep our choice for the sake of readability.

For dimensionality reduction, we assume that both $\hat{\boldsymbol{k}}$ and $\boldsymbol{y} - \boldsymbol{x}$ are projected using the dimensionality-reduction matrix $\boldsymbol{\Phi}$, so the test statistic is the one introduced in Equation (3.11).

Under hypothesis H_1, we are interested in computing the mean and standard deviation of r, and find conditions on $\boldsymbol{\Phi}$ that maximize the Signal-to-Noise Ratio (SNR) defined as $\mathrm{SNR} = \mathrm{E}\{r\} / \sqrt{\mathrm{Var}\{r\}}$.

The mean term is

$$\mathrm{E}\{r\} = \mu_x \mathrm{E}\{\mathrm{Tr}[\boldsymbol{\Phi}_q \boldsymbol{k}_0 \boldsymbol{k}_0^T \boldsymbol{\Phi}_c^T]\} = \mu_x \sigma_k^2 \mathrm{Tr}[\boldsymbol{M}] \tag{3.13}$$

where $\boldsymbol{\Phi}_q \doteq \boldsymbol{\Phi} \boldsymbol{H}_q$, $\boldsymbol{\Phi}_c \doteq \boldsymbol{\Phi} \boldsymbol{H}_c$, and $\boldsymbol{M} \doteq \boldsymbol{\Phi}_q^T \boldsymbol{\Phi}_c$. We assume that projection vectors are normalized, so that each row of both $\boldsymbol{\Phi}_q$ and $\boldsymbol{\Phi}_c$ has l_2-norm equal to 1.

For the variance term, we show in [128] that

$$\begin{aligned} \mathrm{Var}\{r\} = \sigma_k^4 \Big(& \mu_x^2 \mathrm{Tr}[\boldsymbol{M}\boldsymbol{M}] + (\mu_x^2 + \sigma_x^2) \mathrm{Tr}[\boldsymbol{M}\boldsymbol{M}^T] \\ & + \sum_{i,j} r_{\tilde{x}}[i-j] \left(M_{i,i} M_{j,j} + M_{i,j} M_{j,i} \right) \Big) + \sigma_t^2 \sigma_k^2 L \\ & + \sigma_e^2 \sigma_k^2 (\mu_x^2 + \sigma_x^2) L + \sigma_e^2 \sigma_t^2 \mathrm{Tr}[(\boldsymbol{\Phi}\boldsymbol{\Phi}^T)^2] \end{aligned} \tag{3.14}$$

Recalling that $\mathrm{Tr}[\boldsymbol{\Phi}\boldsymbol{\Phi}^T] = L$, the last summand is minimized when $\boldsymbol{\Phi}\boldsymbol{\Phi}^T = \boldsymbol{I}$, that is when projection vectors are orthonormal. Minimization of Equation (3.14) with respect to $\boldsymbol{M}$ is cumbersome; however, noticing that in practice μ_x^2 is often larger than σ_x^2, it makes sense to minimize instead $\sigma_k^4 \mu_x^2 \left(\mathrm{Tr}[\boldsymbol{M}\boldsymbol{M}] + \mathrm{Tr}[\boldsymbol{M}\boldsymbol{M}^T] \right)$, subject to $\mathrm{Tr}[\boldsymbol{M}] = L$.

Writing $\boldsymbol{M} = \boldsymbol{M}_S + \boldsymbol{M}_A$, where $\boldsymbol{M}_S$ and $\boldsymbol{M}_A$ are, respectively, symmetric and anti-

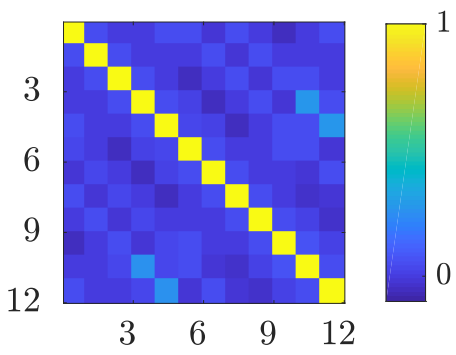


Figure 3.2:

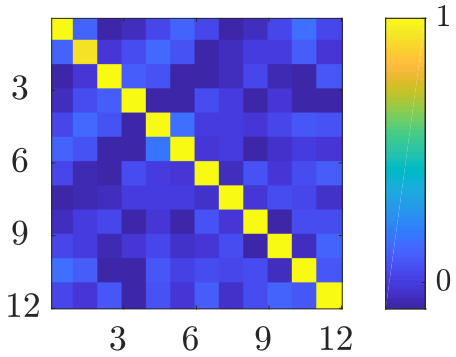


Figure 3.3:

Figure 3.4: $\boldsymbol{\Phi}_c\boldsymbol{\Phi}_q^T$ for $\boldsymbol{\Phi}$ is a Sub-Wrapping matrix (a) and when $\boldsymbol{\Phi}$ is a Gaussian matrix (b) in a small-scale example with $n = 15$, $L = 12$, $s = 3$ and real case $\boldsymbol{H}_c$ and $\boldsymbol{H}_q$.

symmetric, the problem can be formulated as

$$\begin{aligned} \text{minimize} \quad & \text{Tr}[\boldsymbol{M}\boldsymbol{M}_S] \\ \text{subject to} \quad & \text{Tr}[\boldsymbol{M} + \boldsymbol{M}^T] = 2\text{Tr}[\boldsymbol{M}_S] = 2L \end{aligned}$$

Observe that $\text{Tr}[\boldsymbol{M}\boldsymbol{M}_S] = \text{Tr}[(\boldsymbol{M}_S + \boldsymbol{M}_A)\boldsymbol{M}_S] = \text{Tr}[\boldsymbol{M}_S\boldsymbol{M}_S]$. Then, we can pose the problem equivalently in terms of $\boldsymbol{M}_S$. In fact, if $\boldsymbol{\Lambda}$ is the diagonal matrix containing the eigenvalues of $\boldsymbol{M}_S$, the problem becomes

$$\begin{aligned} \text{minimize} \quad & \text{Tr}[\boldsymbol{\Lambda}^2] \\ \text{subject to} \quad & \text{Tr}[\boldsymbol{\Lambda}] = L \end{aligned} \tag{3.15}$$

From Cauchy-Schwarz inequality, the solution to this problem is achieved when all P non-null eigenvalues of $\boldsymbol{M}_S$ are identical and take the value L/P. Notice that $P = \text{rank}(\boldsymbol{M}_S) \leq 2{\cdot}\text{rank}(\boldsymbol{M}) = 2L$. It is an ongoing problem to translate this general solution into design principles regarding $\boldsymbol{\Phi}_c$ and $\boldsymbol{\Phi}_q$. A particular approach consists in assuming that $\boldsymbol{M}$ is symmetric, in which case P above will be equal to L, and the eigenvalues of $\boldsymbol{\Phi}_c\boldsymbol{\Phi}_q^T$ will be all identical to 1. Therefore, we will seek projection matrices $\boldsymbol{\Phi}$ for which $\boldsymbol{\Phi}_c\boldsymbol{\Phi}_q^T = \boldsymbol{I}$.

On the other hand, for hypothesis H_0 we find that $\text{E}\{r\} = 0$ and in [128] we show that $\text{Var}\{r\} = (\mu_x^2 + \sigma_x^2)\text{Tr}[\boldsymbol{M}\boldsymbol{M}^T]$. Then, minimizing $\text{Tr}[\boldsymbol{M}\boldsymbol{M}^T]$ also minimizes the variance for H_0.

As shown above, a condition to nearly maximize detection while minimizing false-alarms in a cross-correlation test is $\boldsymbol{\Phi}_c\boldsymbol{\Phi}_q^T = \boldsymbol{I}$. Since our goal is to design $\boldsymbol{\Phi}$, we first want to find the support of the point-spread function characterizing $\boldsymbol{H}_c$ and $\boldsymbol{H}_q$.

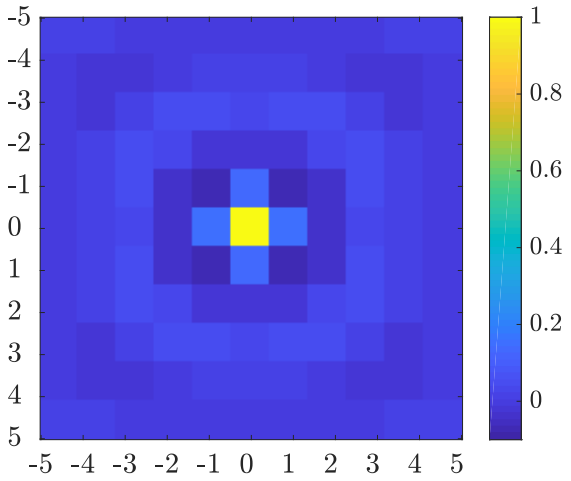


Figure 3.5: Estimated camera PRNU $\hat{\boldsymbol{K}}_c$ autocorrelation matrix as space-invariant bi-dimensional linear filter with $h = 11$ in a small-scale example with $N = 225$.

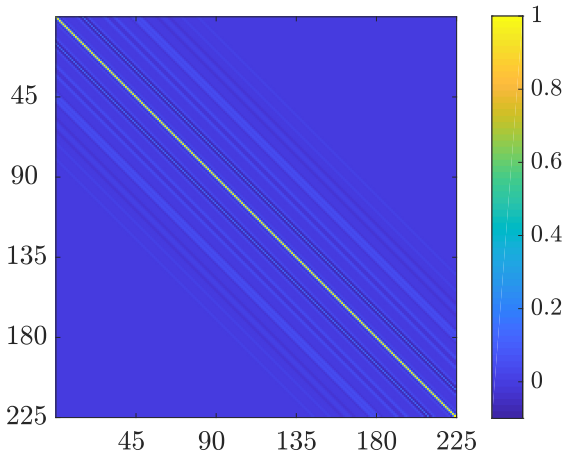


Figure 3.6: $\boldsymbol{H}_c$ approximation as space-invariant bi-dimensional linear filter with $h = 11$ in a small-scale example with $N = 225$.

To this end, we consider a bi-dimensional estimated camera PRNU fingerprint $\hat{\boldsymbol{K}}_c$ and compute its autocorrelation, as shown in Figure 3.5. If the estimated fingerprint were purely a white noise signal, its autocorrelation would result in a single peak at $(0, 0)$. However, demosaicing and low-pass filtering occurring during image compression lead the autocorrelation to be far from an impulse, thus with a support extended over more than one pixel. Approximating the PRNU autocorrelation as a bi-dimensional point-spread function with support $h \times h$, we can estimate $\boldsymbol{H}_c$ as a space-invariant bi-dimensional linear filter, as from the example in Figure 3.6. Following the same reasoning, it is possible to estimate the support of $\boldsymbol{H}_q$ via autocorrelation of $\boldsymbol{W}_q$, the bi-dimensional residual for

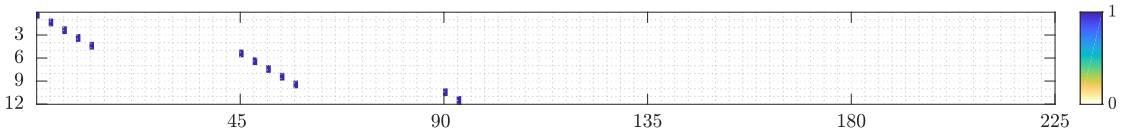


Figure 3.7:

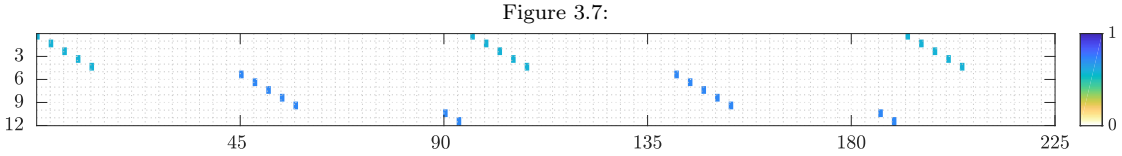


Figure 3.8:

Figure 3.9: Projection matrix $\boldsymbol{\Phi}$ designed as a bi-dimensional Sub-Sampling matrix $\boldsymbol{\Phi}^{ss}$ (a) and the derived Sub-Wrapping matrix $\boldsymbol{\Phi}^{sw}$ (b) in a small-scale example with $n = 15$, $L = 12$, $s = 3$.

image $\boldsymbol{Y}_q$. For the sake of simplicity, in the following we will consider $\boldsymbol{H}_q = \boldsymbol{H}_c$. The extension to the case where they are different is straightforward.

When it comes to the design of $\boldsymbol{\Phi}$, among the infinite set of solutions that lead to $\boldsymbol{\Phi}_c \boldsymbol{\Phi}_q^T = \boldsymbol{I}$, we propose two simple design strategies: i) Sub-Sampling; ii) Sub-Wrapping.

Sub-Sampling The first proposed strategy is bi-dimensional subsampling with step-size s, cropped to the first L output coefficients. In order to build the projection matrix $\boldsymbol{\Phi}$ let us suppose we wish to project a bi-dimensional fingerprint $\hat{\boldsymbol{K}}_c \in \mathbb{R}^{n \times n}$, with $N = n^2$, L and n integer multiples of step-size s, such that $n = b \cdot s$, for some $b \in \mathbb{Z}$. Let us define

$$\begin{aligned}
\boldsymbol{a} &\doteq \left[\begin{array}{cccc} 1 & 0 & \dots & 0 \end{array} \right] \in \{0,1\}^s \\
\boldsymbol{B} &\doteq \left[\begin{array}{c|c} \boldsymbol{I}_{b \times b} & \boldsymbol{0}_{b \times b(s-1)} \end{array} \right] \in \{0,1\}^{b \times bs} \\
\bar{\boldsymbol{\Phi}}^{ss} &\doteq \boldsymbol{I}_{b \times b} \otimes (\boldsymbol{B} \otimes \boldsymbol{a}) \in \{0,1\}^{b^2 \times N}
\end{aligned}$$

where $\boldsymbol{a}$ is a row vector of s elements, representing the structuring element for the subsampling matrix, $\boldsymbol{I}_{b \times b}$ is an identity matrix of size $b \times b$, $\boldsymbol{0}_{b \times b(s-1)}$ is an all-zero matrix of size $b \times b(s-1)$ and $\otimes$ represents the Kronecker product between matrices. The subsampling projection matrix $\boldsymbol{\Phi}^{ss} \in \{0,1\}^{L \times N}$ is then obtained by retaining the first L rows of $\bar{\boldsymbol{\Phi}}^{ss}$. An example for matrix $\boldsymbol{\Phi}^{ss}$ is shown in Figure 3.7.

Sub-Wrapping The second strategy is an extension of subsampling with step-size s that takes into account the unused input coefficients due to cropping to length L. Cropped rows from $\bar{\boldsymbol{\Phi}}^{ss}$ are warped and summed to obtain $\boldsymbol{\Phi}^{sw}$, taking care of normalizing each

row of $\boldsymbol{\Phi}^{sw}$. Formally, let us define

$$\delta \doteq Ls^2, \quad r \doteq \left\lceil \frac{b^2}{L} \right\rceil$$
$$\boldsymbol{Z}_{i,j}^{[z]} = \begin{cases} 1 & \text{if } i + z = j \\ 0 & \text{otherwise} \end{cases} \qquad i \in [1, N],\ j \in [1, N]$$
$$\boldsymbol{D} \doteq \sum_{z=0}^{r-1} \boldsymbol{Z}^{[z\delta]} \in \{0, 1\}^{N \times N}$$

where δ is the offset, in input space, between successive warps, r is the maximum number of input samples contributing to a single output sample and $\boldsymbol{D}$ is an offset projection matrix. We can now define an over-sized projection matrix $\bar{\boldsymbol{\Phi}}^{sw}$ as

$$\bar{\boldsymbol{\Phi}}^{sw} \doteq \bar{\boldsymbol{\Phi}}^{ss} \boldsymbol{D} \in \{0, 1\}^{b^2 \times N}$$

and finally derive the actual subwrapping projection matrix $\boldsymbol{\Phi}^{sw} \in \mathbb{R}^{L \times N}$ by selecting the first L rows of $\bar{\boldsymbol{\Phi}}^{sw}$ and normalizing each row in l_2 norm. An example of the resulting projection matrix $\boldsymbol{\Phi}^{sw}$ is depicted in Figure 3.8.

Orthogonality The design of $\boldsymbol{\Phi}^{ss}$and $\boldsymbol{\Phi}^{sw}$ presents two degrees of freedom, namely, s and $L \leq N/s^2$. When designing a Sub-Sampling or a Sub-Wrapping projection matrix, the rule that allows us to meet the $\boldsymbol{\Phi}_c \boldsymbol{\Phi}_q^T = \boldsymbol{I}$ requirement is $s \geq h$, since this way replicas of the point-spread function will not overlap in $\boldsymbol{\Phi}_c$ or $\boldsymbol{\Phi}_q$.

Figure 3.4 shows an example of how Sub-Wrapping and Gaussian Random Projections affect $\boldsymbol{\Phi}_c \boldsymbol{\Phi}_q^T$ when real-world (i.e. with unlimited support) $\boldsymbol{H}_c$ and $\boldsymbol{H}_q$ are considered. With the same set of parameters of the figure ($n = 15, L = 12, s = 3$), the resulting values for $\mathrm{Tr}[\boldsymbol{\Lambda}^2]$ are 14.58, 14.92, 16.88 respectively for Sub-Sampling, Sub-Wrapping, Gaussian Random Projection. Recalling from Equation (3.15) that the objective function to minimize is $\mathrm{Tr}[\boldsymbol{\Lambda}^2]$, we can see how Gaussian Random Projection, despite being built as an orthogonal basis, presents a higher value for $\mathrm{Tr}[\boldsymbol{\Lambda}^2]$ than Sub-Sampling and Sub-Wrapping, which leads to a worst expected performance, as we will confirm later.

Complexity A final consideration about the design of $\boldsymbol{\Phi}$ is in terms of complexity. In the Sub-Sampling case, projection complexity is reduced only to samples selection, as no sum or multiplications are involved, thus $\mathcal{C}^{ss} = 1$. For the Sub-Wrapping case, sample selection is followed by summing of a maximum of r elements for each output sample, thus $\mathcal{C}^{sw} = L(r-1)$. When $\boldsymbol{\Phi}$ is built as a circulant matrix, as for the Gaussian Random

Projections case, $\mathcal{C}^{RP} = 2N[2\log_2(N) + 3]$, considering a direct Fast-Fourier-Transform (FFT) of size N, a dot product between two complex vectors of length N and the final inverse FFT of size N. In a real-case application with $n \approx 1,500$, $s = 3$, $b \approx 500$, $L \approx 150,000$ and $r = 2$, the complexity of Sub-Wrapping projection is $\mathcal{C}^{sw} \approx 150$K, less than 0.1% with respect to the complexity of Gaussian Random Projections $\mathcal{C}^{RP} \approx 200$M.

3.4.3 Random-projections-based pipeline design

Algorithm 2 Camera fingerprint processing algorithm

Require: $\hat{\boldsymbol{K}}$, d, $\boldsymbol{\phi}$, P, δ
$\hat{\boldsymbol{K}}_{\text{tmp}} \leftarrow$ DECIMATE($\hat{\boldsymbol{K}}$, d)
$\check{\boldsymbol{K}} \leftarrow$ DECIMATE($\hat{\boldsymbol{K}}_{\text{tmp}}$, d)
$\check{\boldsymbol{k}} \leftarrow$ UNWRAP($\check{\boldsymbol{K}}$)
$\boldsymbol{r}^* \leftarrow$ RANDOM PROJECT($\check{\boldsymbol{k}}$, $\boldsymbol{\phi}$, P)
$\boldsymbol{r}^\delta \leftarrow$ DEAD-ZONE QUANTIZE($\boldsymbol{r}$, δ)
bit-stream $\leftarrow$ ENTROPY ENCODE($\boldsymbol{r}^\delta$)

function DECIMATE($\mathbf{A}$, d)
  $h, w \leftarrow$ SIZE($\mathbf{A}$)
  for i_r in $\{0, 1, \ldots, h-1\}$ **do**
    for i_c in $\{0, 1, \ldots, \lfloor (w-1)/d \rfloor\}$ **do**
      $A_d(i_r, i_c) = 0$
      for j_c in $\{0, 1, \ldots, w-1\}$ **do**
        $x = i_c \cdot d - j_c$
        $A_d(i_r, i_c) += h_c(x) A(i_r, j_c)$
      end for
    end for
  end for
  return $\mathbf{A}_d$
end function

function RANDOM PROJECT($\mathbf{A}$, $\boldsymbol{\phi}$, P)
  $\mathbf{a} \leftarrow$ FLATTEN($\mathbf{A}$)
  $\mathbf{a}^{\mathcal{F}} \leftarrow \text{FFT}(\mathbf{a}_d)$, $\boldsymbol{\phi}^{\mathcal{F}} \leftarrow \text{FFT}(\Phi)$
  $\mathbf{p}^{\mathcal{F}} \leftarrow \mathbf{a}^{\mathcal{F}} \boldsymbol{\phi}^{\mathcal{F}}$
  $\mathbf{p} \leftarrow \text{IFFT}(\mathbf{p}^{\mathcal{F}})$
  $\mathbf{p} \leftarrow$ TRIM($\mathbf{p}$,P)
  return $\mathbf{p}$
end function

function DEAD-ZONE QUANTIZE($\mathbf{a}$, δ)
  $P \leftarrow$ LENGTH($\mathbf{a}$), $\sigma \leftarrow$ STD($\mathbf{a}$)
  for i in $\{0, \ldots, P-1\}$ **do**
    if $a(i) > \delta\sigma$ **then**
      $a_q(i) = +1$
    else if $a(i) < -\delta\sigma$ **then**
      $a_q(i) = -1$
    else
      $a_q(i) = 0$
    end if
  end for
  return $\mathbf{a}_q$
end function

function ENTROPY ENCODE($\mathbf{a}$)
  $P \leftarrow$ LENGTH($\mathbf{a}$)
  for i in $\{0, \ldots, P-1\}$ **do**
    if $a(i) = 1$ **then**
      bitstream $\leftarrow$ bitstream +10
    else if $a(i) = -1$ **then**
      bitstream $\leftarrow$ bitstream +11
    else
      bitstream $\leftarrow$ bitstream +0
    end if
  end for
  return bitstream
end function

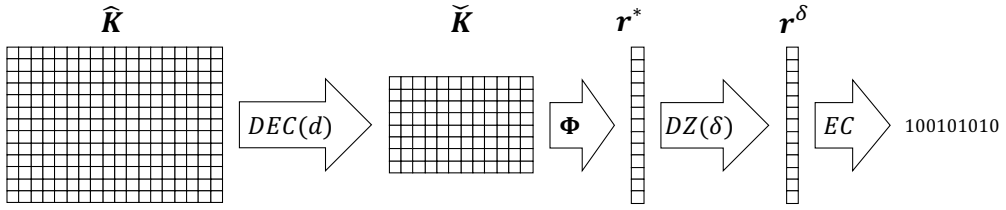


Figure 3.10: Proposed compression pipeline. A camera fingerprint estimate $\hat{\boldsymbol{K}}$ is first decimated (DEC) by a factor d to obtain $\check{\boldsymbol{K}}$, then Random Projections (RP) are used to compress $\check{\boldsymbol{K}}$ into a P elements vector $\boldsymbol{r}^*$. Finally a dead-zone quantizer (DZ) is applied with threshold δ to get a quantized fingerprint $\boldsymbol{r}^\delta$ successively encoded (EC) to generate a compressed output bit-stream.

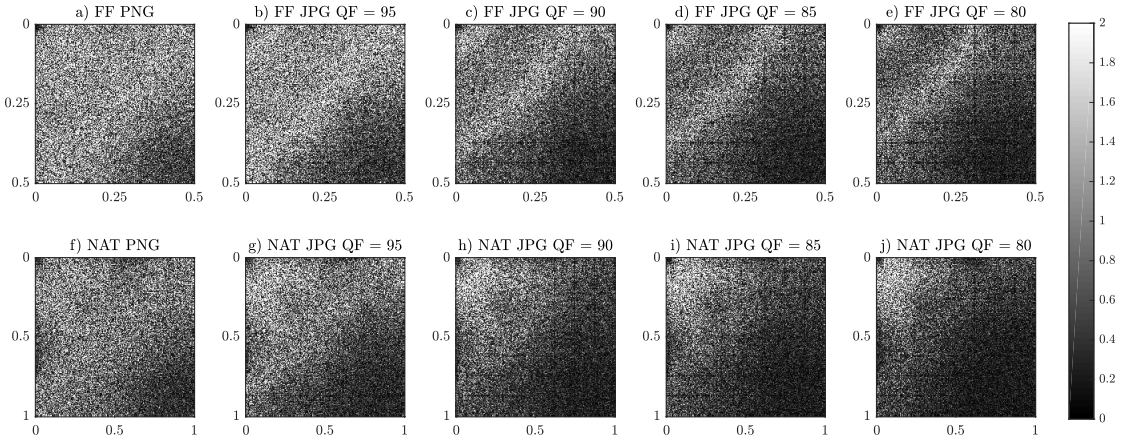


Figure 3.11: Power Spectral Density (PSD) for noise residuals from a single flatfield image (a, b, c, d, e) and from a single natural image (f, g, h, i, j) on PNG uncompressed images and while varying JPEG quality factor.

Given a camera fingerprint $\hat{\boldsymbol{K}}_c$ acquired according to Equation (3.4) and a query residual $\mathbf{W}_q$ we propose the same compression pipeline for both $\hat{\boldsymbol{K}}_c$ and $\mathbf{W}_q$. For the sake of clarity, in the following we describe the process only for $\hat{\boldsymbol{K}}_c$, using $\hat{\boldsymbol{K}}$ as a short notation for $\hat{\boldsymbol{K}}_c$.

Algorithm 2 and Figure 3.10 depict the proposed approach, comprising four steps: i) fingerprint $\hat{\boldsymbol{K}}$ is decimated over rows and columns by a factor d to generate $\hat{\boldsymbol{K}}_d$; ii) Random Projections (RP) are applied to $\hat{\boldsymbol{K}}_d$ to produce a vector $\boldsymbol{r}_P^*$ of length P; iii) dead-zone (DZ) quantization with threshold δ preserves only the peaks of $\boldsymbol{r}_P^*$ and creates $\boldsymbol{r}_P^\delta$; iv) Entropy Coding (EC) applied to $\boldsymbol{r}_P^\delta$ generates the compressed output bit-stream containing fingerprint information. In the following we illustrate the rationale and the details for each step, introducing compression for the case of the fingerprint estimate $\hat{\boldsymbol{K}}$. The whole pipeline holds exactly in the same way also for query residuals $\mathbf{W}_q$.

Decimation It is known from [118] that JPEG compression increases the variance of cross-correlation values in Equation (3.5), thus reducing the margin between H_0 and H_1 hypotheses. In Figure 3.11 we analyze the effect of JPEG compression on the Power Spectral Density (PSD) for different quality factors by looking at noise residuals $\boldsymbol{W}$ extracted from flatfield and natural images. It is clear that the power of the residue at high spatial frequencies – lower right quadrant – lowers as images are more compressed. Moreover, residual PRNU contributions in high-frequency bins are combined with residuals of blockiness artifacts from JPEG compression that cannot be completely removed by the residue extraction process.

Putting together the aforementioned consideration, a reasonable and simple preprocessing method to reduce the dimensionality of $\hat{\boldsymbol{K}}$ and attenuate its high-frequency components consists in decimating $\hat{\boldsymbol{K}}$ by a factor $d > 1$ along rows and columns. This operation is accomplished via interpolation with a cubic kernel [129] $h_c(x)$ defined as

$$h_c(x) = \begin{cases} 1.5|x|^3 - 2.5|x|^2 + 1 & \text{if } |x| \leq 1 \\ -0.5|x|^3 + 2.5|x|^2 - 4|x| + 2 & \text{if } 1 < |x| \leq 2 \\ 0 & \text{otherwise} \end{cases} \tag{3.16}$$

Given a vector $\boldsymbol{a}$ of length L and a decimation factor d, the i-th element of its decimation a_d results in

$$a_d(i) = \sum_{j=0}^{L-1} h_c(j - i \cdot d) \cdot a(j),\ \forall i \in \{0, \ldots, \lfloor l/d \rfloor\} \tag{3.17}$$

The choice of d is carried out such that the resulting resized fingerprint $\check{\boldsymbol{K}}$ performs in the same way as the original $\hat{\boldsymbol{K}}$ fingerprint in terms of detection performance.

Random Projection The second step in the pipeline consists in projecting $\check{\boldsymbol{k}}$ – the column-wise unwrapping of $\check{\boldsymbol{K}}$ – with P Random Projections according to Equation (3.8), to obtain $\boldsymbol{r}^*$. Recalling that $L = w \cdot h$ is the number of pixels in the sensor, it is worth observing that the input to Random Projection is now a vector with L/d^2 elements due to the previous decimation, thus the computational complexity in terms of additions and multiplications of implementing Equation (3.8) for $\check{\boldsymbol{k}}$ is

$$\mathrm{C}_{RP}(\check{\boldsymbol{k}}) = 2\frac{L}{d^2}\left[\log_2(L) + 3 - 2\log_2(d)\right] \tag{3.18}$$

whereas the computational complexity of implementing Equation (3.8) for $\hat{\boldsymbol{k}}$ – the column-wise unwrapping of $\hat{\boldsymbol{K}}$ – as in [15] is

$$\mathrm{C}_{RP}(\hat{\boldsymbol{k}}) = 2L\left[\log_2(L) + 3\right] \tag{3.19}$$

Finally observing that

$$\mathrm{C}_{RP}(\check{\boldsymbol{k}}) < \frac{1}{d^2}\mathrm{C}_{RP}(\hat{\boldsymbol{k}}) \tag{3.20}$$

we can conclude that the computational complexity is reduced by more than a factor d^2.

Dead-zone quantization While binarization of random projections has been proved to be an effective way of quantizing and preserving good performance in terms of detection, here we propose to use a dead-zone quantizer on $\boldsymbol{r}^*$ to get $\boldsymbol{r}^\delta$. Given σ, the standard deviation of $\boldsymbol{r}^*$, the i-th element of $\boldsymbol{r}^\delta$ for $i = 1, \ldots, P$ is obtained as

$$r^\delta(i) = \begin{cases} +1 & \text{if } r^*(i) > \delta\sigma \\ 0 & \text{if } -\delta\sigma \leq r^*(i) \leq \delta\sigma \\ -1 & \text{if } r^*(i) < -\delta\sigma \end{cases} \tag{3.21}$$

The rationale behind this choice is twofold. On one hand, as observed in [50] for PRNU digest, peaks with high absolute values are the most important ones in terms of cross-correlation, thus preserving those peaks seems a reasonable choice. On the other hand, quantizing with a variable threshold allows to reduce the bitrate of $\boldsymbol{r}^\delta$ via entropy coding by fixing P while increasing the value of δ (i.e., setting more samples to zero). When comparing a dead-zone quantized signal with another binarized or dead-zone quantized signal, the similarity measure provided by the cross-correlation, Equation (3.5) is equivalent to the Opposite Absolute Distance (OAD) defined as:

$$\mathrm{OAD}\left(\boldsymbol{x}, \boldsymbol{y}\right) = \sum_{i=1}^{N} 1 - |\boldsymbol{x}(i) - \boldsymbol{y}(i)| \tag{3.22}$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are respectively the two binarized or dead-zone quantized reference fingerprint and query residuals of length N.

Entropy coding As last step of the pipeline, an arithmetic entropy coding scheme is applied to $\boldsymbol{r}^\delta_P$ in order to get a compressed bit-stream containing compressed information about $\hat{\boldsymbol{K}}$. The threshold value δ is a tunable parameter, as will be discussed in Section 3.6. The Gaussian distribution of PRNU fingerprint coefficients after projection comes at help

in the entropy coding stage. In fact, dead-zone quantization produces a three-symbols output whose entropy is lower than $\log_2(3)$ for increasing values of δ, due to the higher probability of symbol 0 with respect to symbols $+1$ and -1.

3.5 Antiforensics

In this section we present the problem of device anonymization, Section 3.5.1, with two different approaches to the topic.

In Section 3.5.2 we present a no-reference technique based on image inpainting.

In Section 3.5.3 we introduce a reference-based technique that explores the use of a double-input CNN auto-encoder for sensor-based antiforensics.

3.5.1 Problem formulation

Given an image $\boldsymbol{Y}$, attenuate its PRNU traces ($\boldsymbol{w}_q$) so that it is impossible to attribute the image to the camera that shot it. In other words, if $\boldsymbol{Y}$ is acquired with a camera whose estimated PRNU is $\boldsymbol{k}_c$, we expect that $\langle \hat{\boldsymbol{k}}_c, \boldsymbol{w}_q \rangle > \tau$. Our goal is to modify $\boldsymbol{Y}$ through editing techniques in order to obtain an anonymized version $\tilde{\boldsymbol{Y}}$ with the following properties:

- the cross-correlation between the noise extracted from the output image $\tilde{\boldsymbol{Y}}$ (i.e., $\tilde{\boldsymbol{w}}_q$) and the camera PRNU gets to reasonably low values (i.e., $\langle \hat{\boldsymbol{k}}_c, \boldsymbol{w}_q \rangle < \tau$)
- $\tilde{\boldsymbol{Y}}$ is not visually corrupted by the applied editing operations (i.e., peak-signal-to-noise ratio between $\boldsymbol{Y}$ and $\tilde{\boldsymbol{Y}}$ assumes high values).

3.5.2 Inpainting-based method

The proposed pipeline, shown in Figure 3.12, is the following: (i) select and delete different blocks of $\boldsymbol{Y}$; (ii) blocks of $\boldsymbol{Y}$ are processed in parallel, reconstructing the erased pixels from neighboring ones as in a classical inpainting problem; (iii) image $\hat{\boldsymbol{Y}}$ is reconstructed by splicing together all the inpainted blocks; (iv) the final anonymized image $\tilde{\boldsymbol{Y}}$ is obtained by further processing pixels around edges of $\hat{\boldsymbol{Y}}$, in order to mitigate inpainting artifacts near sharp discontinuities and increase visual quality. The algorithm is applied separately on each color plane. In the following, we describe each step considering focusing on a single color plane, without loss of generality.

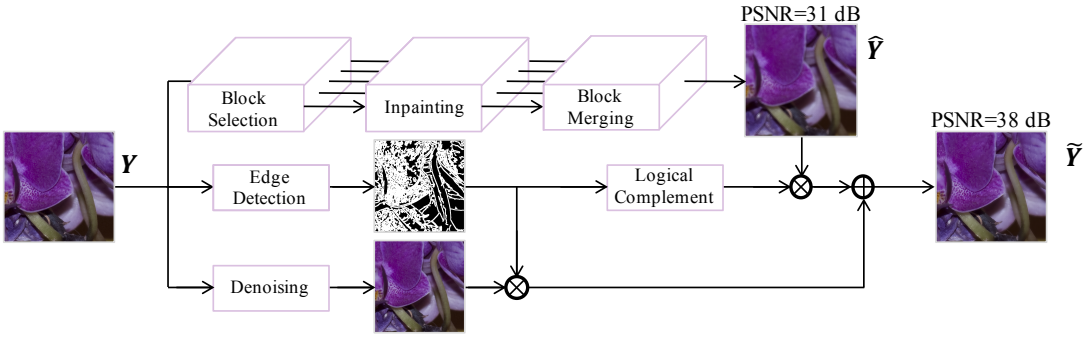


Figure 3.12: Pipeline of the proposed anonymization strategy: an original image $\boldsymbol{Y}$ undergoes block selection, and every modified version is inpainted in parallel. Results are merged to obtain $\hat{\boldsymbol{Y}}$, which is further processed around edges, thus obtaining the anonymized image $\tilde{\boldsymbol{Y}}$.

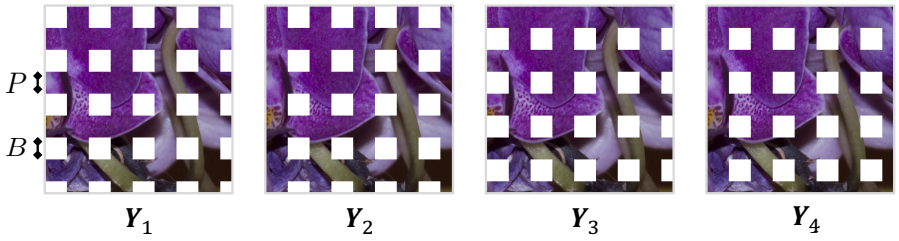


Figure 3.13: Results of applying 4 pixel selectors to image $\boldsymbol{Y}$: each selector S_s selects and deletes different blocks of the original image. White areas represent deleted pixels.

Block selection The first step consists in selecting and deleting different image blocks as shown in Figure 3.13. The selection of pixels is performed through N pixel selectors $\boldsymbol{S}$, $s \in [1, N]$, applied in an element-wise fashion to $\boldsymbol{Y}$. Specifically, each pixel selector $\boldsymbol{S}_s$ is a matrix with the same size of the image, set to 0 if the pixel must be removed and to 1 elsewhere. This matrix, if multiplied pixel-wise with image $\boldsymbol{Y}$, selects and erases areas of $B \times B$ pixels, interleaved in both horizontal and vertical directions by a fixed gap of P pixels left at their original values. In order to gradually select and cancel all image pixels, each selector $\boldsymbol{S}_s$ is orthogonal to the others. This means that each selector deletes a different portion of the image, but the whole image is covered considering the effect of all selectors eventually. Figure 3.13 shows an example in which 4 pixel selectors are multiplied by the original image in order to cancel some specific regions. We define each modified version of $\boldsymbol{Y}$ as $\boldsymbol{Y}_s = \boldsymbol{S}_s \circ \boldsymbol{Y}$, with $s \in [1, N]$.

Inpainting methods We investigate the family of inpainting methods based on the solution of simple ℓ_1 and ℓ_2 regularized inverse problems. Our choice is motivated by the fact that this class of problems has been broadly implemented through standard and

efficient iterative methods [130].

In this scenario, the inpainted image $\hat{\boldsymbol{Y}}_s$ is estimated by solving the minimization problem:

$$\hat{\boldsymbol{Y}}_s = \arg\min_{\bar{\boldsymbol{Y}}_s} \ \|\boldsymbol{S}_s \circ \bar{\boldsymbol{Y}}_s - \boldsymbol{Y}_s\|_{\mathrm{F}}^2 + \mu \ \|R(\bar{\boldsymbol{Y}}_s)\|_p^p \ . \tag{3.23}$$

The pixel selector $\boldsymbol{S}_s$ selects the pixels from $\hat{\boldsymbol{Y}}_s$, $\|\cdot\|_{\mathrm{F}}$ represents the Frobenius norm, μ is the penalty weight associated to the regularizer operator $R(\cdot)$, and $\|\cdot\|_p$ is the entrywise ℓ_p norm. The first term of the objective function (i.e., $\|\boldsymbol{S}_s \circ \bar{\boldsymbol{Y}}_s - \boldsymbol{Y}_s\|_F^2$) is a fitting condition. It basically imposes that the inversion result approximately honors the known samples of the image. The second term (i.e., $\mu \ \|R(\bar{\boldsymbol{Y}}_s)\|_p^p$) is the regularizer, which provides the condition to be respected by the inpainted pixels. Different regularizing operators and norms lead to different inpainting results.

A reasonable regularizer condition consists in imposing some smoothness constraints (i.e., $R(\cdot)$ should be a roughening operator). This ensures inpainted values to be not too dissimilar from neighboring pixels, which is desirable especially in flat regions of natural images (i.e., far from edges). Among the many regularization operators available in the literature, we make use of $R(\cdot) = \sqrt{(D_x(\cdot))^2 + (D_y(\cdot))^2}$, where operations are applied element-wise and $D_x(\cdot)$ and $D_y(\cdot)$ are the horizontal and vertical derivative operators, respectively. For instance, the result of applying $D_x(\cdot)$ to an image $\hat{\boldsymbol{Y}}_s$ is defined as

$$\bar{\boldsymbol{Y}}_{s_x,(i,j)} = \begin{cases} \bar{\boldsymbol{Y}}_{s,(i,j)} - \bar{\boldsymbol{Y}}_{s,(i,j+\Delta)} & j \in [1, H-\Delta] \\ 0 & j \in [H-\Delta+1, H] \end{cases} , \tag{3.24}$$

where $\Delta > 0$ is the desired pixel gap for the derivative calculation, H represents the image height and width, as we are dealing with square images, and (i, j) represent the pixel locations within the image. By setting different gaps Δ, various definitions of derivative can be used.

For what concerns the norm applied to the regularization term, we consider two strategies: (i) a ℓ_2 norm leading to the well-known Tikhonov regularization [131], which is the most widespread technique for inverting ill-conditioned problems; (ii) a ℓ_1 norm strategy, known as Total Variation (TV) regularization [132], which exhibits edge preserving properties. While the ℓ_2 strategy is very simple from an implementation point of view, it is known to introduce an overall smoothing effect that may be undesirable around sharp edges. For this reason we also investigate the ℓ_1 norm applied to the previously defined operator, which is widely used for preserving edges and discontinuities in the final inpainting solution.

Block Merging After obtaining the inpainted versions $\hat{\boldsymbol{Y}}_s$ of each image $\boldsymbol{I}_s$, we construct the image $\hat{\boldsymbol{Y}}$ by merging the inpainted blocks:

$$\hat{\boldsymbol{Y}} = \sum_{s=1}^{N} (1 - \boldsymbol{S}_s)\ \hat{\boldsymbol{Y}}_s. \tag{3.25}$$

Note that each pixel of $\hat{\boldsymbol{Y}}$ is an inpainted pixel, and no original pixels of $\boldsymbol{I}$ are preserved by this operation. Moreover, due to $\boldsymbol{S}_s$ definition, each pixel of $\hat{\boldsymbol{Y}}$ is reconstructed from a single $\hat{\boldsymbol{Y}}_s$.

Edge Processing Inpainting applied with the aforementioned solutions may still introduce some undesirable visual artifacts around edges (the effect can be noticed in Figure 3.12, where image $\hat{\boldsymbol{Y}}$ has low peak signal to noise ratio). Therefore, to increase image visual quality, a further edge processing operation is applied. More specifically, we substitute pixels of $\hat{\boldsymbol{Y}}$ around image edges with pixels coming from a denoised version of $\boldsymbol{Y}$, in order to avoid the reintroduction of too much PRNU information. Indeed, [113] shows that denoising can attenuate PRNU, thus motivating our approach.

Formally, the edge reconstruction pipeline is as follows: (i) the original image $\boldsymbol{Y}$ is denoised using two successive steps of BM3D algorithm (with variance parameter $\sigma = 7$) [133], thus obtaining $\dot{\boldsymbol{Y}}$; (ii) $\boldsymbol{Y}$ edges are extracted using Canny edge detector (with its default parameters in MATLAB®), obtaining a binary edge mask $\boldsymbol{E}$ (the same size of $\boldsymbol{Y}$), which is 1 only at edge locations and 0 elsewhere; (iii) edge-mask $\boldsymbol{E}$ is dilated by means of a disk structural element (with radius of 3 pixels); (iv) the final anonymized image is defined as $\tilde{\boldsymbol{Y}} = \hat{\boldsymbol{Y}} \circ (1 - \boldsymbol{E}) + \dot{\boldsymbol{Y}} \circ \boldsymbol{E}$, where each operation is applied pixel-wise. Figure 3.12 shows the effect of applying edge processing to $\hat{\boldsymbol{Y}}$. Note that peak signal to noise ratio (PSNR) of $\tilde{\boldsymbol{Y}}$ is increased by 7 dB. Moreover, the computational complexity of this step is mainly due to denoising operation, which is almost negligible with respect to the rest of the inpainting pipeline.

3.5.3 Autoencoder-based method

In order to anonymize an image $\boldsymbol{Y}$, we propose an anonymization function $\mathcal{A}(\boldsymbol{Y}, \boldsymbol{K})$ that generates an anonymized version of $\boldsymbol{Y}$, namely $\hat{\boldsymbol{Y}} = \mathcal{A}(\boldsymbol{Y}, \boldsymbol{K})$. The design of $\mathcal{A}$ is such that the Peak Signal-to-Noise Ratio (PSNR) between $\boldsymbol{Y}$ and $\hat{\boldsymbol{Y}}$ is greater than a reference value while the normalized cross-correlation between the noise residual $\hat{\boldsymbol{W}}$ extracted from $\hat{\boldsymbol{Y}}$ and $\boldsymbol{K}$ is minimized. In an optimal case, it will result that $\langle \hat{\boldsymbol{K}}, \hat{\boldsymbol{W}} \rangle < \tau$, so that it is not possible anymore to bind the anonymized image $\hat{\boldsymbol{Y}}$ to its camera device with confidence α.

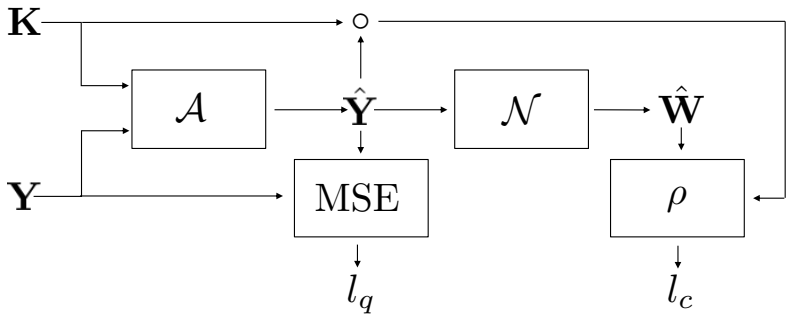


Figure 3.14: Architecture of the proposed system. An anonymization function $\mathcal{A}$ is fed with the input image $\boldsymbol{Y}$ and the relative camera PRNU $\boldsymbol{K}$. The anonymized image $\hat{\boldsymbol{Y}}$ is used to compute a quality loss l_q based on the Mean Squared Error (MSE) between $\hat{\boldsymbol{Y}}$ and $\boldsymbol{Y}$. The noise residual $\hat{\boldsymbol{W}}$, extracted through a noise extraction function $\mathcal{N}$ from $\hat{\boldsymbol{Y}}$, is used together with the camera PRNU $\boldsymbol{K}$ to determine a correlation loss l_c.

The proposed anonymization method is based on the idea of minimizing a cost function made up of two components: i) a measure of the difference between the input image $\boldsymbol{Y}$ and its anonymized version $\hat{\boldsymbol{Y}}$; ii) the cross-correlation between the anonymized noise residual $\hat{\boldsymbol{W}}$ and the camera PRNU $\boldsymbol{K}$.

Figure 3.14 shows the overall working scheme. An image $\boldsymbol{Y}$ and the corresponding camera PRNU $\boldsymbol{K}$ are fed as input to the anonymization function $\mathcal{A}$. The output of $\mathcal{A}$ is an anonymized version of $\boldsymbol{Y}$, namely $\hat{\boldsymbol{Y}}$. The Mean Square Error (MSE) between $\boldsymbol{Y}$ and $\hat{\boldsymbol{Y}}$ is computed and stored into l_q, the first component of the global cost function. The anonymized image $\hat{\boldsymbol{Y}}$ is fed as input to the noise extraction function $\mathcal{N}$ and the output $\hat{\boldsymbol{W}}$ is correlated with the sample-wise product between $\boldsymbol{K}$ and $\hat{\boldsymbol{Y}}$ to get l_c, the second component of the global cost function. The global cost function l is then defined as $l = (1 - \beta) \cdot l_q + \beta \cdot l_c$, where β is a weighting parameter tailored at balancing the trade-off between image quality and anonymization performance.

In the depicted scheme, $\mathcal{N}$ is a fixed noise extractor, whereas $\mathcal{A}$ is a denoising function learned independently on every pair $(\boldsymbol{Y}, \boldsymbol{K})$ provided as input. We require both $\mathcal{N}$ and $\mathcal{A}$ to support gradient computation so that it is possible to learn via back-propagation the parameters of $\mathcal{A}$ as a function of the overall cost function l.

To satisfy the gradient computation capability for $\mathcal{N}$ we resort to DnCNN [120], a fully-convolutional neural network that shows noise extraction capabilities comparable with the Wavelet-based filtering approach commonly used for PRNU-based image source attribution. DnCNN is composed by a set of 17 convolutional layers composed by 64 filters each with kernel size equal to 3 and padding 1, each followed by ReLU non linearity and batch normalization. The fully-convolutional nature of the network does not require as input a fixed size image and produces as output a noise residual with the same size of the input image.

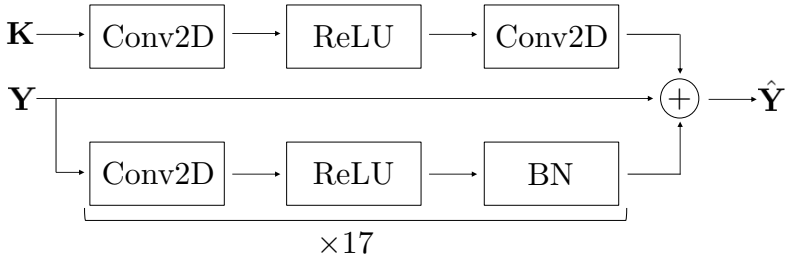


Figure 3.15: Structure of the proposed CNN-based anonymization function $\mathcal{A}$. The input image $\boldsymbol{Y}$ is processed through a set of 17 convolutional layers (Conv2D) followed by ReLU non-linearity and Batch Normalization (BN). The reference PRNU $\boldsymbol{K}$ is processed with two convolutional layers separated by a ReLU non-linearity. The output anonymized image $\hat{\boldsymbol{Y}}$ results from the sample-wise algebraic sum of the input image $\boldsymbol{Y}$ and the two fully-convolutional branches.

Algorithm 3 Image-wise anonymization process

Require: $\boldsymbol{Y}$, $\boldsymbol{K}$, $\mathrm{PSNR}_{\mathrm{min}}$
  $\beta \leftarrow 0.1$
  for i in $\{1, \ldots, 3000\}$ **do**
    $\hat{\boldsymbol{Y}} \leftarrow \mathcal{A}(\boldsymbol{Y}, \boldsymbol{K})$
    $l_q \leftarrow \mathrm{MSE}(\boldsymbol{Y}, \hat{\boldsymbol{Y}})$
    $P \leftarrow \mathrm{PSNR}(\boldsymbol{Y}, \hat{\boldsymbol{Y}})$
    $\hat{\boldsymbol{W}} \leftarrow \mathcal{N}(\hat{\boldsymbol{Y}})$
    $l_c \leftarrow \mid \rho(\hat{\boldsymbol{W}}, \boldsymbol{K} \circ \hat{\boldsymbol{Y}}) \mid$
    $l = (1 - \beta) \cdot l_q + \beta \cdot l_c$
    $\mathcal{A} \leftarrow \textsc{BackPropagate}(\mathcal{A}, l)$
    if MOD(i, 300) $= 0 \wedge P < \mathrm{PSNR}_{\mathrm{min}}$ **then**
      $\beta \leftarrow \beta/4$
    end if
    if $P > \mathrm{PSNR}_{\mathrm{min}} \wedge l_c < 10^{-4}$ **then**
      return $\hat{\boldsymbol{Y}}$
    end if
  end for
  return $\hat{\boldsymbol{Y}}$

As for the choice of $\mathcal{A}$, we exploit an autoencoder structure similar to DnCNN, as depicted in Figure 3.15. The input image $\boldsymbol{Y}$ is processed by a set of 17 convolutional layers (Conv2D), each followed by ReLU non-linearity and batch normalization (BN). The reference PRNU $\boldsymbol{K}$ is fed to a convolutional layer, followed by a ReLU and yet another convolutional layer. The final anonymized image $\hat{\boldsymbol{Y}}$ results from the sum of the two convolutional processing branches together with the input image itself. The weights of the convolutional layers and the parameters of batch normalization for $\mathcal{A}$ are learned for every single image via back-propagation, driven by the global cost function l.

The image-wise anonymization process follows as from Algorithm 3. An input Image $\boldsymbol{Y}$, a reference PRNU $\boldsymbol{K}$ and a minimum desired Peak Signal-to-Noise Ratio ($\mathrm{PSNR}_{\mathrm{min}}$)

$i = 30$	$i = 410$	$i = 970$	$\mathbf{Y}$
$\rho = 0.122$	$\rho = 0.012$	$\rho = 0.004$	
$P = 27.3$	$P = 34.3$	$P = 42.2$	

Figure 3.16: Iterations of the proposed algorithm on a sample image. From left to right the evolution of $\hat{\boldsymbol{Y}}$ at $i = \{30, 310, 970\}$ with cross-correlation ρ decreasing and PSNR P increasing. The rightmost picture is the original image $\boldsymbol{Y}$.

are provided as input. The loss weighting factor β is initialized at 0.1. At every iteration the anonymized image $\hat{\boldsymbol{Y}}$ is first computed, together with the MSE loss l_q and the PSNR P with respect to the original image. Then the noise extraction function $\mathcal{N}$ is used to extract a noise residual $\hat{\boldsymbol{W}}$ from the anonymized image and compute the cross-correlation loss l_c. The global loss l is computed according to the weighting factor β. As all operations in $\mathcal{A}$ are differentiable, it is possible to back-propagate the error and modify $\mathcal{A}$ parameters to minimize loss with any iterative optimization algorithm (e.g., stochastic gradient descent in our implementation). It is not required for $\mathcal{N}$ and $\mathcal{A}$ to have a similar structure, as long as both are differentiable operators. Once every 300 iterations if the PSNR value P is smaller than the desired minimum value PSNR_{min} the weighting factor β is reduced by a factor 4, to raise the importance of the MSE loss l_q over the cross-correlation loss l_c. If the current PSNR is greater than the desired minimum and the cross-correlation loss is small enough (i.e., $l_c < 10^{-4}$), the current anonymized image $\hat{\boldsymbol{Y}}$ is returned and the optimization stops. At most 3000 iterations of the algorithm are performed, in order to bound the required anonymization time if the early stop condition is not met. A sample of the evolution of $\hat{\boldsymbol{Y}}$, ρ and P over the iteration is provided in Figure 3.16.

3.6 Experiments

In this section we reports the experimental setup and the results related to PRNU compression, Section 3.6.1, and PRNU-based antiforensics, Section 3.6.2.

3.6.1 PRNU compression

Setup To validate the effectiveness of the proposed methods, we focus on the problem of image source attribution in a probabilistic framework with the following constraints:

- All extracted fingerprints $\hat{\boldsymbol{K}}_c$ and residuals $\mathbf{W}_q$ are cropped to their central region of size $h = w = 1500$, thus $L = 2.25 \cdot 10^6$ is the number of elements for each vector $\hat{\boldsymbol{k}}_c$ and $\boldsymbol{w}_q$. This allows for a direct comparison between every fingerprint-residual pair.

- We consider only aligned fingerprints and residuals at original resolution, meaning that we are not looking for rotation, cropping, or other affine transformations. All fingerprints and residuals have the same size and for each camera device the cropped region offset with respect to the origin is fixed. This choice follows from state-of-the-art works about PRNU compression [15, 119].

Given a set of C camera devices, for each device $c \in [1, C]$ we have F_c flatfield pictures that we use to estimate the camera fingerprint $\hat{\boldsymbol{K}}_c$, according to Equation (3.4). In this way we build a dataset $\mathcal{C}$ of N_c device fingerprints. For each natural image $\boldsymbol{Y}^{(n)},\ n \in [1, N]$ we estimate its residual $\boldsymbol{W}^{(n)}$, building a dataset $\mathcal{Q}$ of N query images.

Evaluation metrics In order to determine whether a query image residual $\boldsymbol{W}^{(n)}$ from $\mathcal{Q}$ is correctly binded to its camera device and not to other devices, we build a cross-correlation matrix $\mathbf{CC} \in \mathbb{R}^{C \times N}$ defined as

$$\begin{aligned} \mathbf{CC}_{c,n} &= \left\langle \hat{\boldsymbol{K}}_c,\ \boldsymbol{W}^{(n)} \right\rangle \\ c &= 1, \dots, C,\ n = 1, \dots, N \end{aligned} \tag{3.26}$$

then we set the same cross-correlation threshold τ for all camera devices such that the overall False-Positive rate is below a certain false-alarm probability p_{FA}. The cross-correlation matrix $\mathbf{CC}$ is then turned into a binary prediction matrix $\mathbf{P} \in \{0, 1\}^{C \times N}$ according to

$$\begin{aligned} \mathbf{P}_{c,n} &= \begin{cases} 1 & \mathbf{CC}_{c,n} > \tau \\ 0 & \mathbf{CC}_{c,n} \leq \tau \end{cases} \\ c &= 1, \dots, C,\ n = 1, \dots, N \end{aligned} \tag{3.27}$$

Comparison between $\mathbf{P}$ and the Ground-Truth binary matrix $\mathbf{GT}$, where $GT(c, n) = 1$ when $\hat{\boldsymbol{K}}_c$ and $\boldsymbol{W}^{(n)}$ are from the same camera device, leads to definition of the hereinafter used evaluation metrics known as True-Positive Rate (TPR) and False-Positive Rate (FPR). In particular, when evaluating the relationship between residual bitrate and system performance, we are considering the True-Positive Rate at a specific false-alarm probability $p_{\text{FA}} = 0.05$.

Datasets Resorting to images from the Dresden Image Database [94] we build two camera fingerprint datasets and several query residual datasets.

Four controlled compression datasets are built upon RAW images coming from 6 camera devices, two for each model of *Nikon-D200*, *Nikon-D70*, *Nikon-D70s*:

- $\mathcal{C}_f^{\mathrm{RAW}}$ is composed of 6 camera fingerprints extracted from flatfield RAW images.
- $\mathcal{Q}_n^{\mathrm{RAW}}$ is composed of 1317 query residual extracted from natural RAW images.
- $\mathcal{Q}_n^{\mathrm{QF}=q}$ is composed of 1317 query residual extracted from natural RAW images compressed in JPEG format with $\mathrm{QF} = q$ before the noise extraction process.

Two uncontrolled compression datasets are built upon JPEG images from 53 camera models, the same used in [15]:

- $\mathcal{C}_f^{\mathrm{JPG}}$ is a composed of 53 camera fingerprints extracted from flatfield JPEG images as encoded by cameras' firmware.
- $\mathcal{Q}_n^{\mathrm{JPG}}$ is a composed of 9092 query residual extracted from natural JPEG images as encoded by cameras' firmware.

3.6.1.1 Sub-Sampling and Sub-Wrapping

Experiments conducted in this Section aim at proving the compression effectiveness of the proposed PRNU fingerprint and residuals projection methods described in Section 3.4.2. Each reported plot presents four curves, each one related to a different compression pipeline:

- Crop (blue): a central square region of the fingerprint or residual is cropped and the rate is modulated by varying region size. This is a baseline reference method.
- Random Projections [RP] (green): Gaussian Random Projections implemented via circulant sensing matrices are used for fingerprint and residual projection. The rate is modulated by varying the projected subspace dimensionality.
- Sub-Sampling [SS] (purple): Bi-dimensional subsampling is implemented as depicted in Section 3.4.2. The rate is modulated by varying the downsamping step size s while keeping $L = bc$.
- Sub-Wrapping [SW] (red): Bi-dimensional subwrapping with $s = 3$. The rate is modulated by varying L.

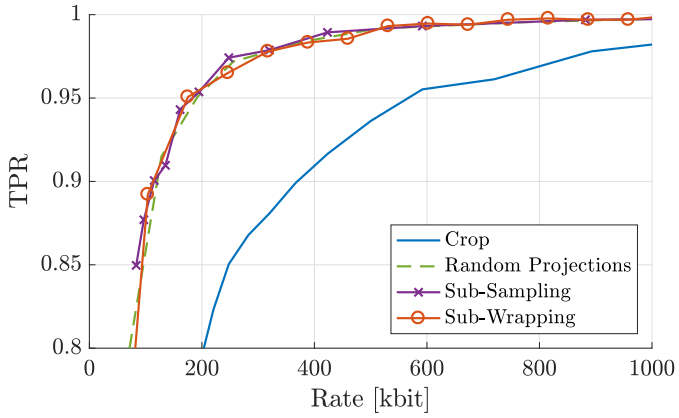


Figure 3.17: Rate vs TPR when camera fingerprints and query residuals are extracted from RAW images. Fingerprints and residuals are not quantized.

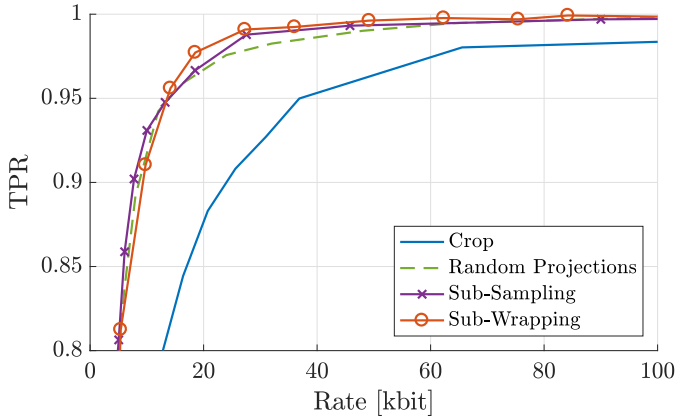


Figure 3.18: Rate vs TPR when camera fingerprints and query residuals are extracted from RAW images. Fingerprints and residuals are binarized after projection.

Figure 3.17 reports results obtained with camera fingerprints from $\mathcal{C}_f^{RAW}$ and query residuals from $\mathcal{Q}_n^{RAW}$. Images used to estimate camera fingerprints and query residuals have been affected only by demosaicing and neither projected fingerprints nor residuals are quantized. We observe how the SS, SW and RP perform at par, a symptom of almost diagonal $\boldsymbol{H}_c$ and $\boldsymbol{H}_q$ matrices due to reduced point-spread function support h.

Figure 3.18 shows how SS and SW behavior changes when both camera fingerprints and query residuals are binarized after projection. Both SS and SW are now outperforming RP with a noticeable compression improvement, e.g. a rate reduction in the order of 37% for a TPR of 0.99.

Figure 3.19 reports results when images used to estimate camera fingerprints and query residuals have undergone JPEG compression by camera built-in firmware. In this case

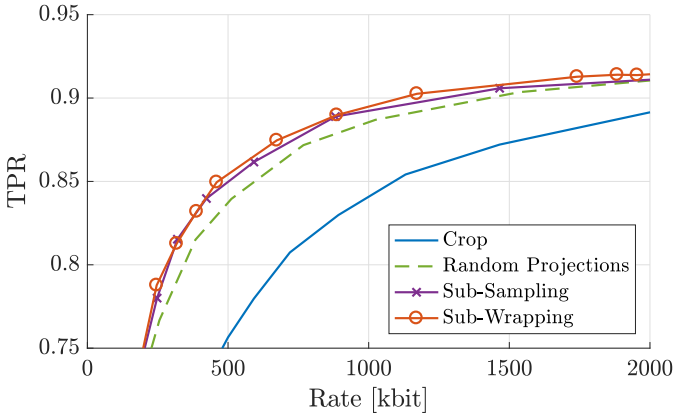


Figure 3.19: Rate vs TPR when camera fingerprints and query residuals are extracted from JPEG images. Fingerprints and residuals are not quantized.

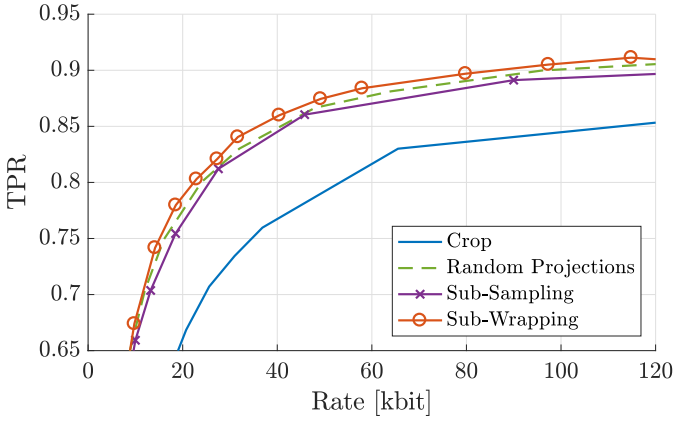


Figure 3.20: Rate vs TPR when camera fingerprints and query residuals are extracted from JPEG images. Fingerprints and residuals are binarized after projection.

the point-spread function in $\boldsymbol{H}_c$ and $\boldsymbol{H}_q$ have surely larger support. This experiment resembles a real-world scenario, where camera fingerprints are taken from $\mathcal{C}_f^{JPG}$, query residuals are from $\mathcal{Q}_n^{JPG}$ and no quantization is applied to projected fingerprints and residuals. In this case SS is performing as RP, with better TPR for bitrates smaller than 1700kbit per residual. SW is instead always offering a TPR improvement in the order of 1% meaning a rate reduction of nearly 23% at a fixed TPR of 0.9.

Finally, Figure 3.20 shows how binarization affects the aforementioned real case scenario on $\mathcal{C}_f^{JPG}$ and $\mathcal{Q}_f^{JPG}$. While with similar performance, among the three methods SW achieves better detection performances at equal bitrate with respect to RP and SS.

In any case, we once again remark that the complexity of the new projection methods is much lower than that of RP.

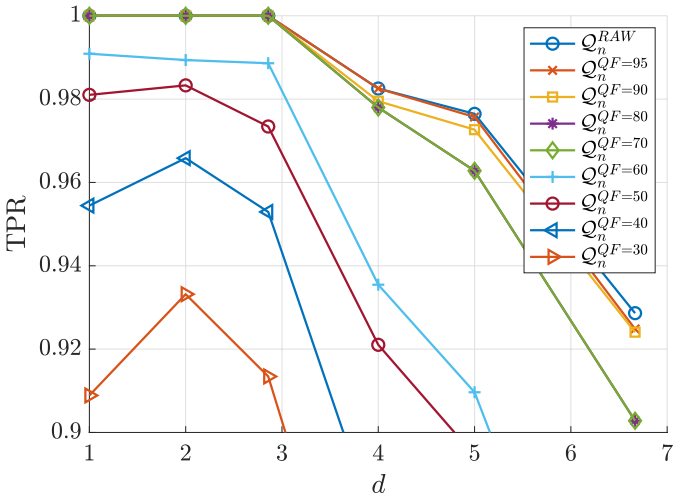


Figure 3.21: Effect of decimating by factor d in terms of detection performance when query images are uncompressed ($\mathcal{Q}_n^{\mathrm{RAW}}$) or JPEG compressed with different quality factors ($\mathcal{Q}_n^{\mathrm{QF}=q}$, $q \in \{30, 40, 50, 60, 70, 80, 95\}$).

3.6.1.2 Random-projections-based pipeline

In the following we report the experimental results related to the compression pipeline proposed in Section 3.4.3.

At first we show how to properly select the decimation kernel and factor. Then we compare Random Projections with and without the proposed resizing and dead-zone quantization approaches in terms of bitrate vs. True-Positive Rate. Finally we show how the proposed pipeline compares with the state-of-the-art solution in terms of Receiver-Operating-Characteristic.

Decimation Choice of d, the resizing factor for the first step of the pipeline, is performed evaluating the impact in terms of TPR when database fingerprints are extracted from $\mathcal{C}_f^{\mathrm{RAW}}$, while query residuals are extracted from $\mathcal{Q}_n^{\mathrm{RAW}}$ and $\mathcal{Q}_n^{\mathrm{QF}=q}$, $q \in \{30, 40, 50, 60, 70, 80, 95\}$. Figure 3.21 depicts the TPR at fixed $p_{\mathrm{FA}} = 0.05$, as a function of d. For weak JPEG compression (QF $\geq$ 70) we observe a drop in detection performance when decimating with a factor $d \geq 3$, while for $d < 3$ the accuracy is preserved almost without loss at a value of 1.0. For stronger JPEG compression factors (QF $<$ 70) decimation with $d = 2$ results beneficial, as it increases the Signal-to-Noise Ratio between the PRNU (signal) and the PRNU-unrelated noise components remaining after the noise extraction process. It is also interesting to notice that the loss-less behavior of decimation with $d = 2$ might be related to CFA interpolation, even though we have no experimental evidences to prove it at this time. Given the aforementioned considerations we chose to

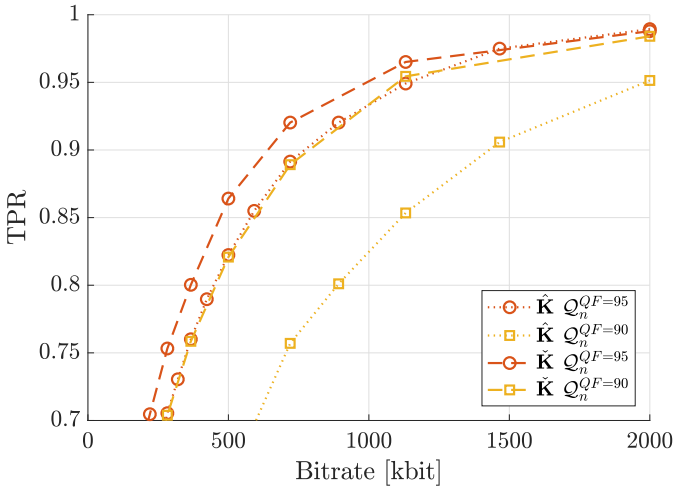


Figure 3.22: Comparison between decimation with a factor $d = 2$ followed by central cropping ($\check{\boldsymbol{K}}$) against central cropping alone ($\hat{\boldsymbol{K}}$) in terms of detection performance when query images are JPEG compressed with different quality factors ($\mathcal{Q}_n^{\mathrm{QF}=95}$, $\mathcal{Q}_n^{\mathrm{QF}=90}$).

set $d = 2$ for all the following experiments. As shown in Section 3.4.3 this leads to a 75% complexity reduction in terms of subsequent Random Projections. As for the choice of the kernel function, the cubic one defined in Equation (3.16) shows similar detection rates when compared to Lanczos kernels, with a noticeable improvement with respect to a bilinear kernel, as it should be expected.

To understand the effect introduced by decimation when dealing with JPEG compressed query images, Figure 3.22 reports the comparison between a baseline central cropping strategy ($\hat{\boldsymbol{K}}$ dotted lines) versus a compression approach based on decimation by a fixed factor $d = 2$ followed by central cropping ($\check{\boldsymbol{K}}$ dashed lines). To vary the bitrate when no decimation is applied ($\hat{\boldsymbol{K}}$ dotted lines) we centrally crop both the fingerprint and the residual. When decimation of a fixed factor $d = 2$ is applied as a pre-processing step ($\check{\boldsymbol{K}}$ dashed lines) the bitrate is varied by central cropping both the decimated fingerprint and the decimated residual. The bitrate is computed as $l^2 \cdot 32\text{bit}$, where l is the side-length of the cropping square. Database images are drawn from $\mathcal{C}_f^{\mathrm{RAW}}$ while query images are extracted from $\mathcal{Q}_n^{\mathrm{QF}=95}$ and $\mathcal{Q}_n^{\mathrm{QF}=90}$. Results show that when JPEG query images are involved the same TPR can be obtained with a significantly lower bitrate, meaning that the interpolation effect introduced by the cubic kernel is preserving PRNU components and compacting them into a smaller support.

When Random Projections are used instead of central cropping, the benefits of decimation are confirmed and highlighted. Figure 3.23 shows the benefit of decimation with $d = 2$ when Random Projections are used to compress the signal while varying the pro-

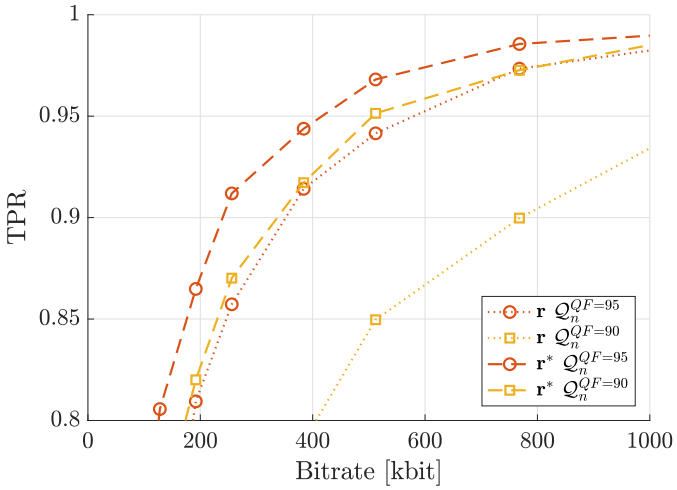


Figure 3.23: Effect of decimation with a factor $d = 2$ followed by Random Projections in terms of detection performance when query images are JPEG compressed with different quality factors ($\mathcal{Q}_n^{\text{QF}=95}$, $\mathcal{Q}_n^{\text{QF}=90}$).

Table 3.1: Entropy coded query rate [kbit] @ TPR = 95% with several quantizer choices in joint and query compression scenarios. Best results in bold font.

P	**Quantization**	**Joint QF 95**	**Joint QF 90**	**Query QF 95**	**Query QF 90**
96k	Dead-zone, $\delta\sigma$	**22**	**30**	**13**	**18**
96k	Dead-zone, δ	44	54	35	42
varying	Binarization	30	39	19	26
varying	Uniform scalar, 3 levels	28	40	18	26
varying	Uniform scalar, 5 levels	33	45	23	31
varying	Uniform scalar, 7 levels	39	52	27	34
varying	Lloyd-Max scalar, 3 levels	34	46	22	33
varying	Lloyd-Max scalar, 5 levels	44	62	30	41
varying	Lloyd-Max scalar, 7 levels	53	68	34	46

jection space dimensionality P. As no quantization is involved, the bitrate is computed as $P \cdot 32$bit. The comparison between the use of Random Projections applied directly to the input fingerprint or residual ($\boldsymbol{r}$ dotted lines) against the use of Random Projections after decimation ($\boldsymbol{r}^*$ dashed lines) show that in the latter case the same TPR is obtained with a significant reduction of bitrate.

Quantization and coding Figure 3.24 shows the reduction in terms of bitrate at equal TPR when dead-zone quantization is used instead of binarization for compressing query residuals. Camera fingerprints extracted from the $\mathcal{C}_f^{\text{RAW}}$ database are decimated with $d = 2$, projected with Random Projection with $P = 96k$ and binarized. Query residuals

Table 3.2: Query rate [kbit] @ TPR = 95% with different encoders in joint and query compression scenarios. AC = Arithmetic Coding, RLC = Run-Length Coding

P	Quantization	Encoding	Joint QF 95	Joint QF 90	Query QF 95	Query QF 90
96k	Dead-zone	AC	22	30	13	18
96k	Dead-zone	RLC	28	40	18	23

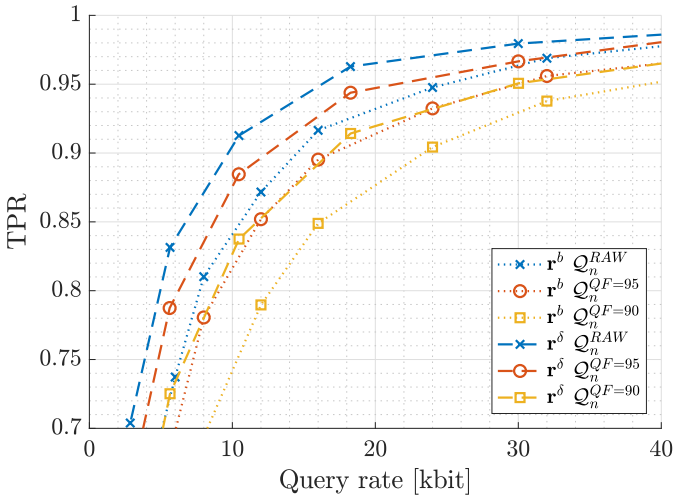


Figure 3.24: Effect of dead-zone quantization, applied after decimation and Random Projection, in terms of detection performance when query images are uncompressed ($\mathcal{Q}_n^{\text{RAW}}$) or JPEG compressed with different quality factors ($\mathcal{Q}_n^{\text{QF=95}}$, $\mathcal{Q}_n^{\text{QF=90}}$).

from $\mathcal{Q}_n^{\text{RAW}}$, $\mathcal{Q}_n^{\text{QF=95}}$ and $\mathcal{Q}_n^{\text{QF=90}}$ are first decimated with $d = 2$ then projected with Random Projection with varying P and binarized ($\boldsymbol{r}^b$ dotted lines) or projected with $P = 96k$ and quantized with a varying δ dead-zone quantizer ($\boldsymbol{r}^\delta$ dashed lines). The reported results show how for both uncompressed and JPEG compressed query images, the same TPR can be obtained with a bitrate reduction of more than 20% when using dead-zone quantization.

To confirm the choice of a dead-zone quantizer whose dead-zone is driven by the standard deviation σ of the residual, as described in Section 3.4.3, we also tested several different quantizers followed by an entropy coder and reported the results in Table 3.1. We evaluate the required query rate to reach a TPR of 95%. In the first two lines, a Random Projection with $96k$ output coefficients is fed to two different dead-zone quantizers, the top one with a σ-driven dead-zone and the second one with a signal independent dead-zone. In both cases, the values for δ are the same for all camera devices and a varying value of δ is used to draw a ROC curve. From the ROC curve we derive the bitrate needed to reach a 95% TPR. The other lines of the table are obtained by projecting the decimated

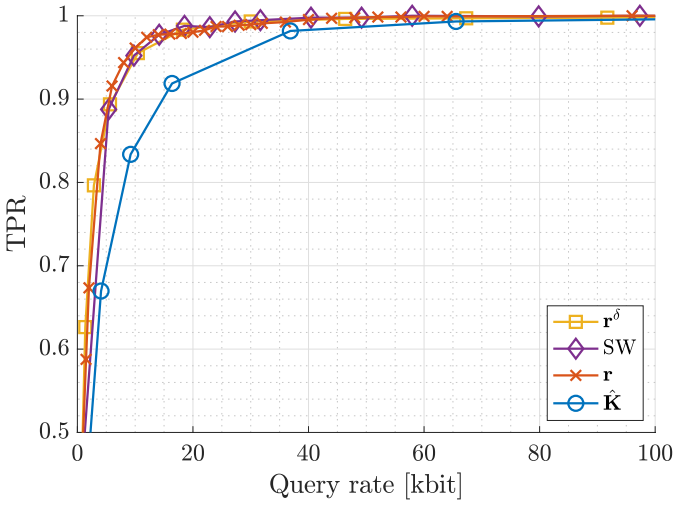


Figure 3.25: Detection performance in a query compression scenario on uncompressed query images.

residuals with a varying projection length P while quantizing with binarization, three uniform scalar quantizers and three Lloyd-Max scalar quantizers. The overall results from the table confirm the choice of a signal-dependent dead-zone quantizer as it reduces the required bitrate for fixed TPR performance.

As final step of the pipeline, the choice of a proper encoding scheme is essential to exploit the reduced entropy resulting from the dead-zone quantization. While results reported in the plots are computed with the use of a real arithmetic encoder, Table 3.2 shows the comparison between an arithmetic coder (AC) applied after dead-zone quantization (first row) compared to run-length coding (RLC) after dead-zone quantization (second row). Run-length coding is obtained by encoding only differential positions and sign of the peaks. The increased bitrate when using RLC is in any case smaller or equal to the bitrate obtained with binarized Random Projections applied to the original query residual (third row of Table 3.1). In applications where bitrate constraints are relaxed, the choice of a run-length encoder allows to keep coding complexity at bay while preserving state-of-the-art compression rates.

Query compression scenario In a query compression scenario we wish to evaluate the trade-off between query residual bitrate and achieved True-Positive Rate. Three different datasets combinations are taken into account, all resorting to camera fingerprints from $\mathcal{C}_f^{\text{RAW}}$ while query residuals are drawn from $\mathcal{Q}_n^{\text{RAW}}$ (Figure 3.25), $\mathcal{Q}_n^{\text{QF=95}}$ (Figure 3.26) and $\mathcal{Q}_n^{\text{QF=90}}$ (Figure 3.27).

Each plot reports four curves comparing different methods: i) central fingerprint and

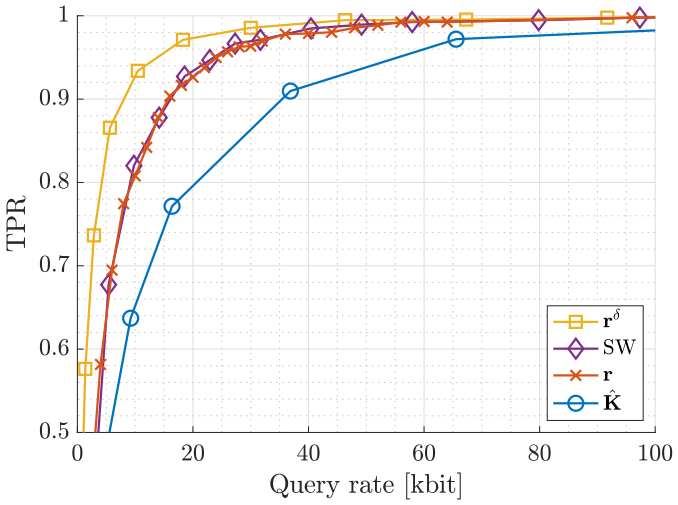


Figure 3.26: Detection performance in a query compression scenario on JPEG compressed query images (QF = 95).

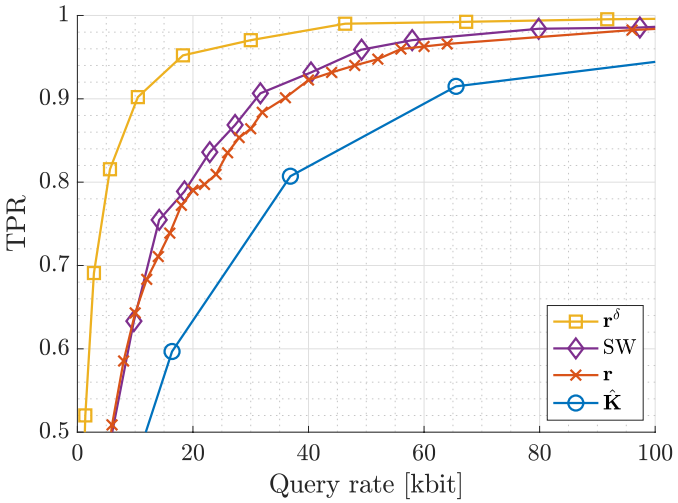


Figure 3.27: Detection performance in a query compression scenario on JPEG compressed query images (QF = 90).

residual cropping ($\hat{\boldsymbol{K}}$) while varying the amount of preserved pixels. Query residual coefficients are quantized by binarization; ii) Random Projections ($\boldsymbol{r}$) applied to the entire $\hat{\boldsymbol{K}}$ fingerprint while varying the number of projection components P. Projected coefficients from query residuals are quantized by binarization; iii) "Sub-Wrapping" method introduced in Section 3.4.2 (SW), which proved to behave at par with Random Projections with a lower computational complexity; iv) proposed method, with $\hat{\boldsymbol{K}}$ resized by a factor $d = 2$ that is then projected through Random Projections with $P = 96000$, giving

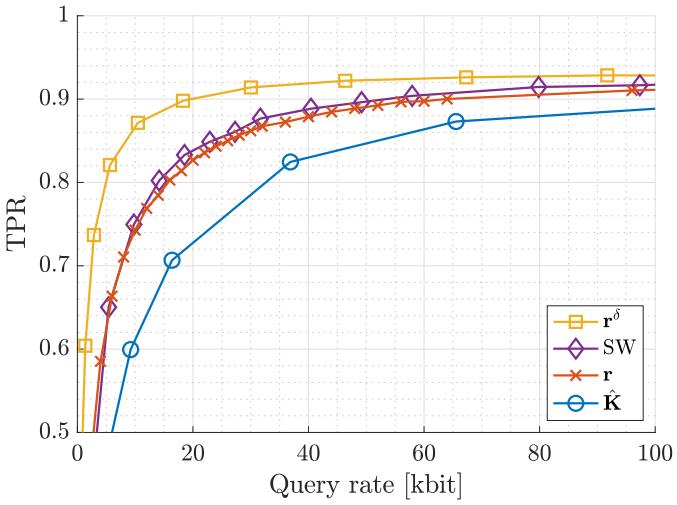


Figure 3.28: Detection performance in a query compression scenario on Dresden dataset.

rise to $\boldsymbol{r}_{96k}$. Query projected fingerprints are then quantized with a Dead-Zone quantizer ($\boldsymbol{r}^\delta$), where threshold δ is gradually increased to decrease query residual bitrate, thanks to arithmetic coding exploiting the reduced entropy of quantized residual.

By observing the three plots we can clearly see that when query residuals are extracted from uncompressed images (Figure 3.25) the performance gap between Random Projections applied to the entire fingerprint ($\boldsymbol{r}$) and dead-zone quantized Random Projections applied to resized fingerprints ($\boldsymbol{r}^\delta$) is negligible. When JPEG compression is applied to query images (Figure 3.26 and Figure 3.27) the gap between $\boldsymbol{r}$ and $\boldsymbol{r}^\delta$ increases greatly. If setting a goal TPR at around 95% the rate required by $\boldsymbol{r}$ is 25kbit and 52kbit, respectively for $\mathcal{Q}_n^{\mathrm{QF}=95}$ and $\mathcal{Q}_n^{\mathrm{QF}=90}$, while for $\boldsymbol{r}^\delta$ it is 13kbit and 18kbit, with a rate reduction between 48% and 65%.

To test performance of proposed method on a dataset with uncontrolled JPEG compression, Figure 3.28 reports results when camera fingerprints are from $\mathcal{C}_f^{JPG}$ and query residuals from $\mathcal{Q}_n^{JPG}$. In this case both camera and query images have undergone JPEG compression, but with several quality factors and customized quantization matrices, due to different brands' firmware implementations. In spite of the uncontrolled condition, when setting a desired TPR at 90% the proposed method achieves a 70% rate reduction with respect to Random Projections.

Joint database and query compression scenario In a joint database and query compression scenario, experiments are carried out while quantizing by binarization all database fingerprints. Figure 3.29, Figure 3.30 and Figure 3.31 report results obtained

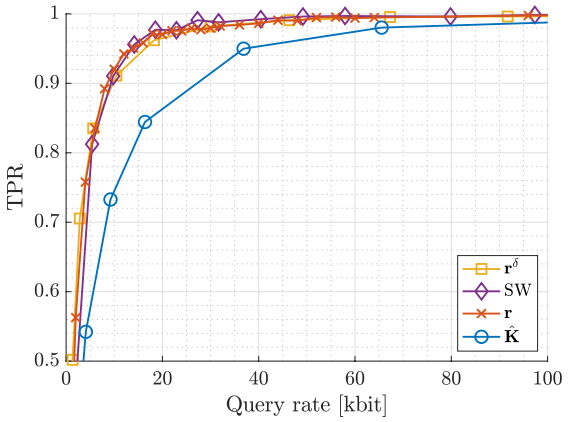


Figure 3.29: Detection performance in a joint compression scenario on uncompressed query images.

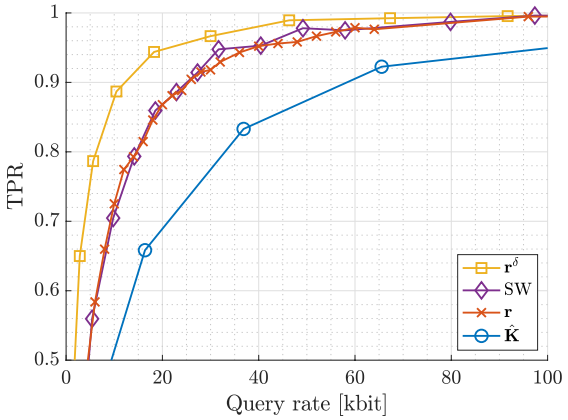


Figure 3.30: Detection performance in a joint compression scenario on JPEG compressed query images (QF = 95).

respectively on query datasets $\mathcal{Q}_n^{\text{RAW}}$, $\mathcal{Q}_n^{\text{QF=95}}$ and $\mathcal{Q}_n^{\text{QF=90}}$. For each plot, the four lines represent performance with same query compression methods illustrated for query compression scenario. When query residuals are extracted from uncompressed images (Figure 3.29) the gap between $\boldsymbol{r}$ and $\boldsymbol{r}^\delta$ results negligible, while as soon as JPEG compression is applied to query images (Figure 3.30 and Figure 3.31) the rate reduction obtained by $\boldsymbol{r}^\delta$ with respect to $\boldsymbol{r}$ is respectively of 46% and 66%, at a desired 95% TPR.

As previously done for the query compression scenario, also in the joint compression scenario we wish to verify performance under uncontrolled JPEG compression. Figure 3.32 reports results obtained from database fingerprints $\mathcal{C}_f^{JPG}$ and query residuals $\mathcal{Q}_n^{JPG}$. As for the query compression scenario, the rate reduction offered by the proposed method $\boldsymbol{r}^\delta$ with respect to Random Projection directly applied to $\hat{\boldsymbol{K}}$ is more than 68%, under a fixed 90% TPR.

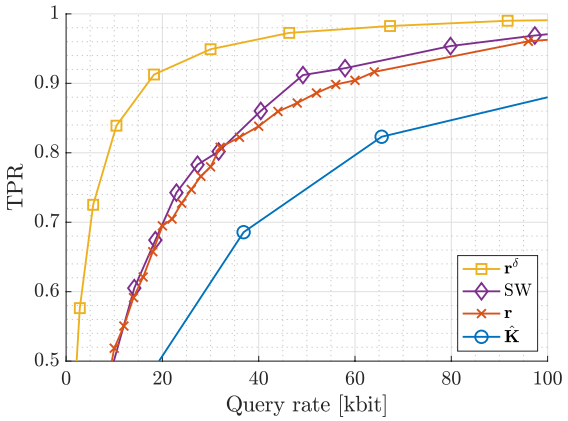


Figure 3.31: Detection performance in a joint compression scenario on JPEG compressed query images (QF = 90).

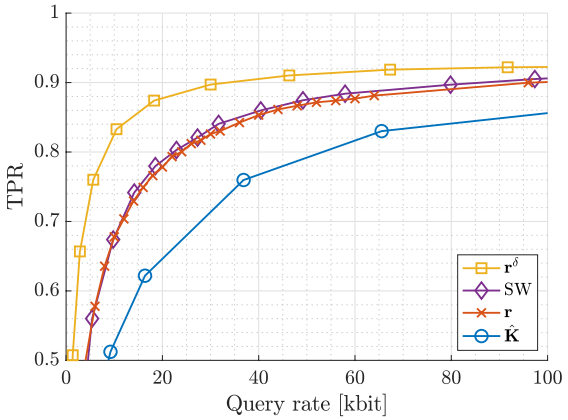


Figure 3.32: Detection performance in a joint compression scenario on Dresden dataset.

Both in the query compression scenario (Figure 3.25,Figure 3.26,Figure 3.27) and in the joint compression scenario (Figure 3.29,Figure 3.30,Figure 3.31) we can observe a common trend. The proposed method ($\boldsymbol{r}^\delta$) performs better in terms of compression than the state of the art method based on solely binarized Gaussian Random Projections ($\boldsymbol{r}$) when query images are JPEG compressed. However, when query images are uncompressed, the proposed method performs at par with the state of the art in terms of compression, but with a reduced computational complexity.

Lowering false-alarm probability A last experiment is carried out by computing the Receiver-Operating-Characteristic (ROC) for both scenarios, query and joint compression, on the Dresden dataset. This test allows us to verify the performance of the proposed pipeline even at p_{FA} smaller than 0.05. With a fixed rate of 64kbit per query for all three

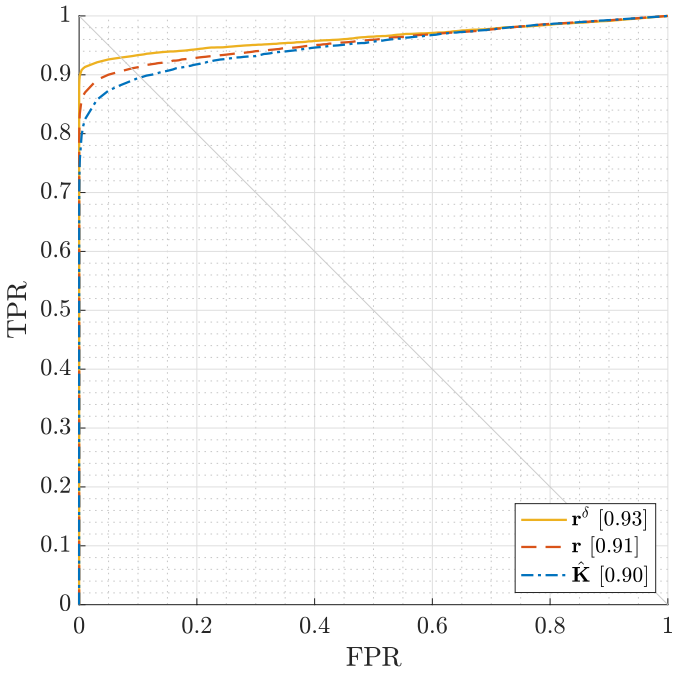


Figure 3.33: Receiver-Operating-Characteristic at 64kbit per residual in a query compression scenario on Dresden dataset.

compared compression methods, Figure 3.33 reports the obtained ROC curves for the query compression scenario, showing the Equal-Error-Rate for each curve at side of legend items. The same results are shown for the joint compression scenario in Figure 3.34. The proposed compression method preserves its good performance even at really small p_{FA}, making this choice viable also for those kind of systems that need to strictly bound the False Positive rate.

Running times The execution time for the query compression pipeline is measured on a modern laptop equipped with a quad-core Intel Core-i7 processor on top of a MATLAB® 2018a implementation. The baseline pipeline that takes as input the image and directly applies Gaussian Random Projections followed by binarization and encoding takes 150ms. When decimation of a factor $d = 2$ is pre-pended to the same pipeline the running time drops to 38ms. Finally, as for the proposed method, when binarization is substituted by dead-zone quantization the total time required to execute the pipeline is 39ms.

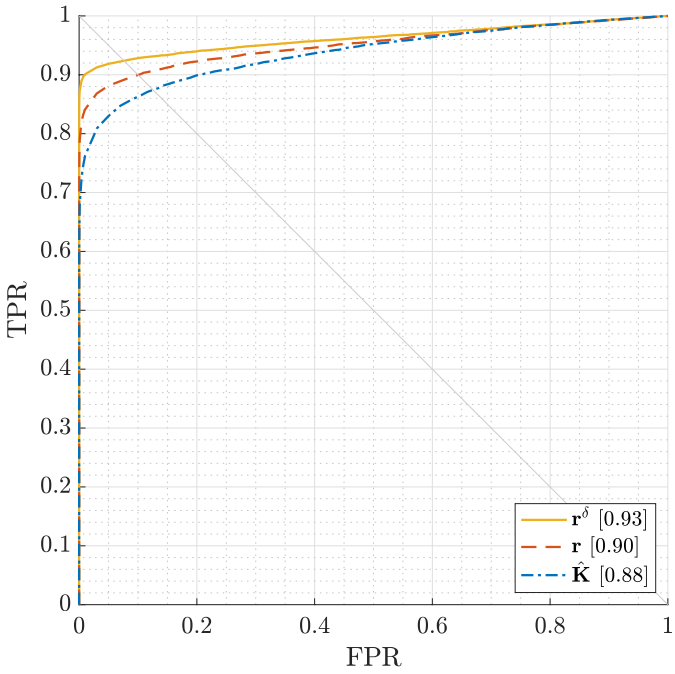
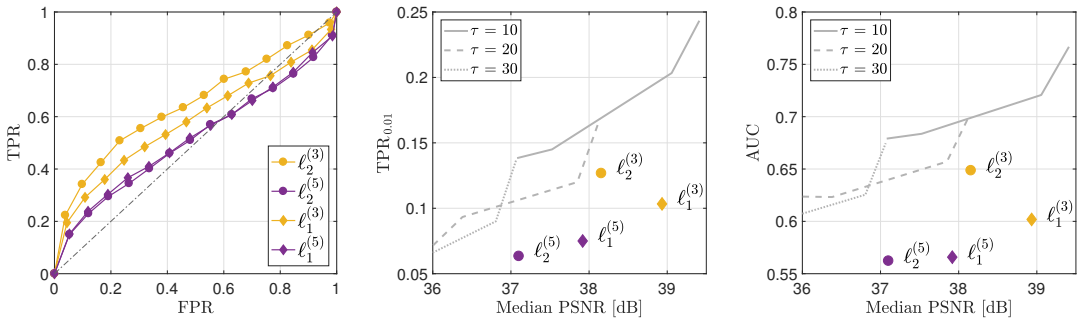


Figure 3.34: Receiver-Operating-Characteristic at 64kbit per residual in a joint compression scenario on Dresden dataset.

3.6.2 PRNU-based anonymization

Setup To state the effectiveness of the proposed approaches, we resort to the same dataset and metrics used in [20]. The dataset is composed of 600 raw natural images, demosaicked with Adobe Lightroom, randomly selected from 6 cameras (Nikon D70, Nikon D70s, Nikon D200, two devices each) from the Dresden Image Database [94]. All the images are cropped in their center to a fixed size of 512×512 pixels.

The estimation of the clean sensor fingerprint $\boldsymbol{K}$ for each camera was obtained from 25 homogeneously lit flatfield images as typically suggested [105, 107]. We evaluate the anonymization performance by using two different noise extraction functions: i) the DnCNN function used as noise extractor within the anonymization scheme, denoted as $\mathcal{N}_{\text{dn}}$; ii) the Wavelet-based noise exaction function [121] commonly used in PRNU-based works, denoted as $\mathcal{N}_{\text{wl}}$. As for the use of DnCNN as noise extractor, we resort to the pre-trained model available from [120]. We resort to Pytorch [134] as Deep Learning and CNN framework.



(a) (b) (c)

Figure 3.35: (a) ROC curves after camera anonymization process (each color represents a different inpainting strategy). (b) Median PSNR vs. $\text{TPR}_{0.01}$ for different inpainting strategies, in comparison with PatchMatch-based (gray lines) patch replacement [20]. (c) Median PSNR vs. AUC for different inpainting strategies, in comparison with PatchMatch-based (gray lines) patch replacement [20].

3.6.2.1 Inpainting-based methods

We show results in terms of receiver operating characteristic (ROC) curves of a camera identification PRNU-based detector. Specifically, fixing the PRNU of a given camera, cross-correlations obtained from anonymized images taken with that camera define the set of positive samples, whereas the set of negative ones includes cross-correlation values from all images not taken with that camera. Our goal is to reduce as much as possible the area under the curve (AUC), thus making the PRNU-based detector not working. Moreover, to evaluate image quality level, we report the relationship between PSNR and true positive rate (TPR) calculated at a fixed false positive rate (FPR) of 1%, which we denote as $\text{TPR}_{0.01}$. The goal is to reach a high PSNR with low values of TPR and AUC. Results are always averaged on all six devices. For the inpainting-based methods we resort to the Wavelet noise extractor for both camera fingerprints and query residuals.

To distinguish between the proposed inpainting strategies, we use the following notation: each technique is identified by the label $\ell_p^{(B)}$, where ℓ_p represents the selected norm for the cost function regularization (i.e., $p \in \{1, 2\}$), while the superscript defines the size of the $B \times B$ regions deleted by the pixel selectors. For what concerns the derivative implementation, we noticed that $\Delta = 3$ provides a good trade off between high reconstruction quality and low PRNU correlation. Indeed, smaller Δ tend to inpaint the image from nearer (and more correlated) pixels, consequently enhancing the correlation with the camera fingerprint. Conversely, larger Δ result in low PSNR. The penalty weight associated to the regularizer is $\mu = 10^{-20}$ for both ℓ_2 and ℓ_1 norms, since higher values oversmooth the image, thus reducing the quality.

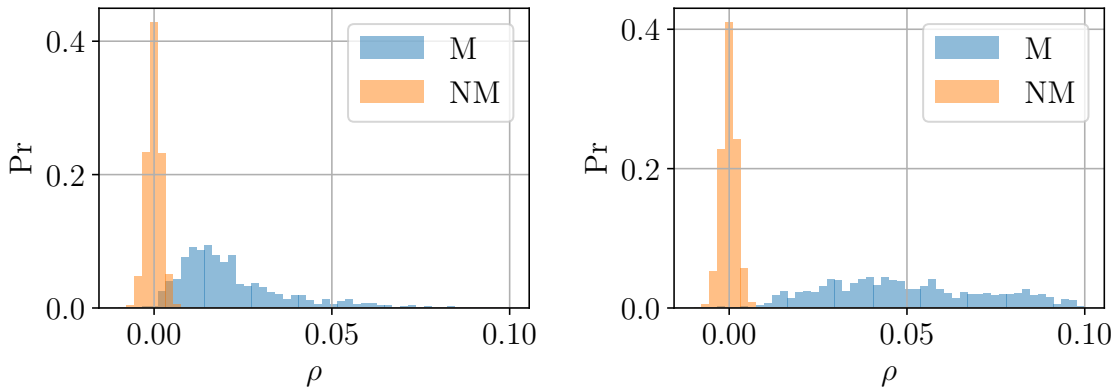
The first experiment aims at showing that the proposed pipeline is actually able to fool PRNU-based camera attribution detectors. To this purpose, we tested both ℓ_2 and ℓ_1 inpainting by considering different block sizes $B \in \{3, 5\}$. Figure 3.35(a) reports promising ROC curves, as their slopes are approximately 45^{o} degrees (i.e., perfect anonymization). Every strategy seems to provide satisfying performances, even though, the bigger the block dimensions ($B = 5$ instead of 3), the better the anonymization results.

For what concerns the inpainting effect on image quality,Figure 3.35(b) and Figure 3.35(c) show encouraging results, both in terms of $\text{TPR}_{0.01}$ and AUC as functions of the median PSNR of inpainted images. As the goal of image anonymization is to reduce TPR or approaching AUC $\simeq 0.5$ still granting high PSNR, the best solutions forFigure 3.35(b) andFigure 3.35(c) are those in the lower right quadrant of the graph. In particular, notice that with the ℓ_1 strategy we are able to reduce $\text{TPR}_{0.01}$ under 11% and to obtain an AUC of 0.6 by still achieving a median PSNR around 39 dB.

As state-of-the-art comparison,Figure 3.35(b) and Figure 3.35(c) also show results of the PatchMatch-based image anonymization proposed in [20], adapted to work on color images. This algorithm depends on two parameters: (i) τ is an error threshold; (ii) σ is a smoothing factor. Each gray line inFigure 3.35(b) andFigure 3.35(c) represents results obtained for a given τ and changing $\sigma \in \{0.1, 0.75, 2, 4\}$. This comparison shows that the proposed methodology is an effective alternative to PatchMatch-based anonymization, both in terms of $\text{TPR}_{0.01}$ and AUC. For instance, the ℓ_1 solution for both $B \in \{3, 5\}$ is able to match state-of-the-art performances in terms of median PSNR, while gaining about 0.1 in terms of both $\text{TPR}_{0.01}$ and AUC.

3.6.2.2 Autoencoder-based method

All the 600 images are anonymized by varying the PSNR_{min} parameter in the set of values $\{37, 38, 39, 40, 41\}$. Each anonymized image is stored as an uncompressed PNG file, thus being quantized to 8-bit as in real case scenario. For each value of PSNR_{min} we observe the distribution of the obtained PSNR values. Noise residuals are extracted with $\mathcal{N}_{\text{dn}}$ and $\mathcal{N}_{\text{wl}}$ for each anonymized image and than correlated with the 6 camera PRNUs. For each PSNR_{min} we compute a Receiver-Operating-Characteristic (ROC) by varying the value of τ, the threshold used in the cross-correlation test to detect an image as being shot from a specific camera. From each ROC we extract both the True-Positive Rate value at a False Alarm probability $\alpha = 0.01$ (TPR@0.01) and the Area Under Curve (AUC). Small AUC values indicate good anonymization performance. Small TPR@0.01 values indicate that when accepting a small false-alarm probability it is not possible to bind the picture to its camera device.



(a) $\mathcal{N}_{\text{dn}}$ (b) $\mathcal{N}_{\text{wl}}$

Figure 3.36: Distribution of normalized cross-correlation values on pristine images using DnCNN ($\mathcal{N}_{\text{dn}}$) and Wavelet ($\mathcal{N}_{\text{wl}}$) noise extractors, for matching image-PRNU pairs (M) and non-matching pairs (NM).

Validation of Denoising Operator First, we need to assess whether DnCNN ($\mathcal{N}_{\text{dn}}$) can be used as a reasonable approximation for the widespread Wavelet ($\mathcal{N}_{\text{wl}}$) noise extractor tailored to PRNU matching and camera device identification. Figure 3.36 shows the distribution of normalized cross-correlation values (ρ) when $\mathcal{N}_{\text{dn}}$ and $\mathcal{N}_{\text{wl}}$ are used as noise extractors from pristine images. In both cases the reference PRNU ($\boldsymbol{K}$) is computed with the Wavelet filter. We can notice that for both noise extractors the discriminability between matching (M) and non-matching (NM) image-camera pairs is preserved, with a slight superimposition of the two distribution for DnCNN. Figure 3.37 shows the difference in terms of Receiver-Operating-Characteristic between $\mathcal{N}_{\text{dn}}$ and $\mathcal{N}_{\text{wl}}$ on pristine images. The values of AUC reported in the legend show how DnCNN detection performance are slightly lower than the ones of Wavelet, but still above 0.99. This test confirms that DnCNN is able to extract PRNU-based residual information from images, thus justifying its use within our anonymization pipeline.

Minimum PSNR Requirement As the proposed algorithm uses the PSNR_{min} to enforce the minimum accepted image quality, we are interested in checking with experiments whether this criteria is actually met. In facts, it might happen that the anonymization loop reaches the maximum number of iterations but the PSNR between $\boldsymbol{Y}$ and $\hat{\boldsymbol{Y}}$ is still smaller than PSNR_{min}. Figure 3.38 reports the histograms of PSNR values obtained for various values of PSNR_{min}. It is possible to notice that for every choice of PSNR_{min} the actual values of PSNR are always greater or equal to the minimum bound. This confirms that the proposed iterative method is able to reach convergence in terms of the imposed

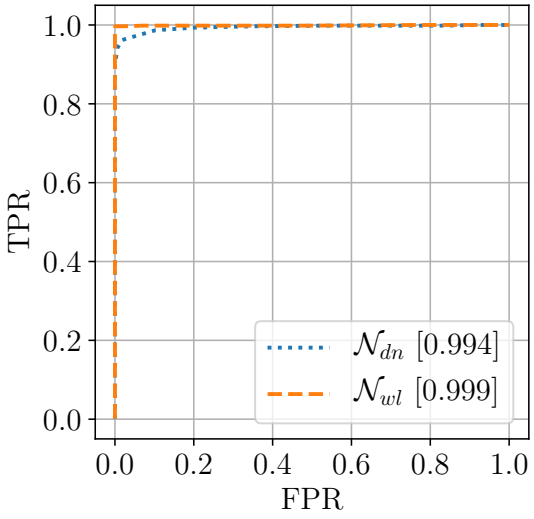


Figure 3.37: Comparison between $\mathcal{N}_{\text{wl}}$ and $\mathcal{N}_{\text{dn}}$ as noise residual extractors in terms of Receiver-Operating-Characteristic. The Area Under Curve reported between squared brackets shows almost equivalent performance in terms of detection.

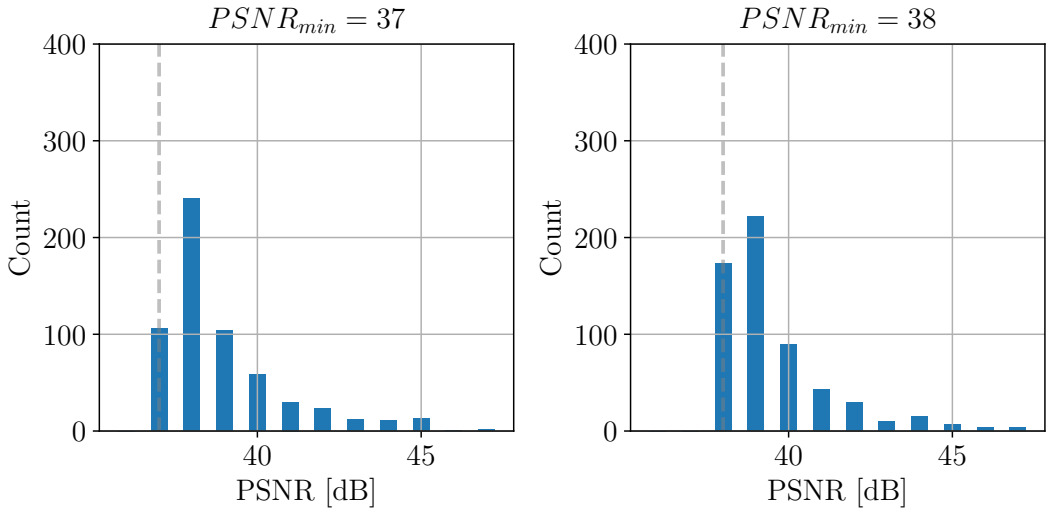
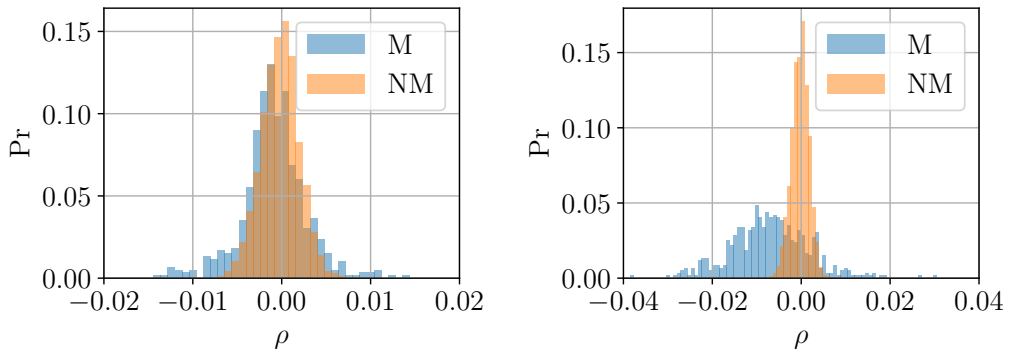


Figure 3.38: Real PSNR distribution when varying PSNR_{min} in $\{37, 38\}$. The real PSNR values are always greater or equal the the minimum value (vertical dashed gray line). The same behavior is obtained for different PSNR_{min} values.

minimum PSNR requirement.

Image Anonymization When it comes to verify the effectiveness of the proposed pipeline in reducing PRNU-based device identification, we first compute the distribution of matching and non-matching normalized cross-correlation (ρ) values obtained from



(a) $\mathcal{N}_{\text{dn}}$ (b) $\mathcal{N}_{\text{wl}}$

Figure 3.39: Distribution of normalized cross-correlation values on anonymized images using DnCNN ($\mathcal{N}_{\text{dn}}$) and Wavelet ($\mathcal{N}_{\text{wl}}$) noise extractors, for matching image-PRNU pairs (M) and non-matching pairs (NM).

anonymized images with noise residuals extracted with DnCNN ($\mathcal{N}_{\text{dn}}$) and Wavelet ($\mathcal{N}_{\text{wl}}$). Figure 3.39a shows how the distributions of matching and non-matching ρ values, obtained when noise residuals are extracted from $\hat{\boldsymbol{Y}}$ through $\mathcal{N}_{\text{dn}}$, are superimposed. This makes practically impossible to bind an anonymized images to the device it comes from. This means that the proposed anonymization pipeline is working in the proper way, thus it has minimized the cross-correlation between the reference PRNU $\boldsymbol{K}$ and the noise residual extracted through $\mathcal{N}_{\text{dn}}$. As we wish to evaluate the effect of the proposed method when the Wavelet-based noise extractor is used on $\hat{\boldsymbol{Y}}$, Figure 3.39b shows the distribution of matching and non-matching ρ values obtained when noise residuals are extracted with $\mathcal{N}_{\text{wl}}$. We can immediately spot two differences with respect to the $\mathcal{N}_{\text{dn}}$ extractor: i) the mean of the matching values is not anymore zero, but it is shifted toward negative values; ii) the variance of matching cross-correlations is way higher than the variance of non-matching cross-correlations. A forensic investigator acting in a blind way, without the knowledge of the proposed anonymization pipeline, might use the cross-correlation test definition at Equation (3.5) to assess whether an image $\hat{\boldsymbol{Y}}$ under investigation comes from a camera whose PRNU is $\boldsymbol{K}$. However, an attack-aware investigator would also perform another test, evaluating the absolute value of the normalized cross-correlation, thus building a symmetric test $|\langle \boldsymbol{W}, \boldsymbol{K} \rangle| > \tau$. In the plots, we refer to the results obtained with the standard Wavelet detector with $\mathcal{N}_{\text{wl}}$, while the results obtained with the Wavelet symmetric detector are denoted as $\mathcal{N}^{\text{a}}_{\text{wl}}$.

Figure 3.40 shows the ROC on anonymized images detection for $\text{PSNR}_{\text{min}} = 40$. If $\mathcal{N}_{\text{dn}}$ is used to extract the noise residual from $\hat{\boldsymbol{Y}}$ we get almost perfect anonymization performance. This confirms that the anonymization loop, based on the minimization of

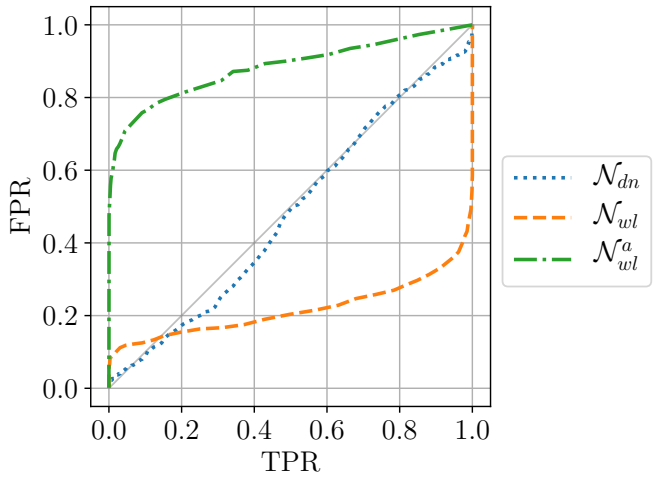


Figure 3.40: Receiver-Operating-Characteristic for $\text{PSNR}_{\text{min}} = 40$.

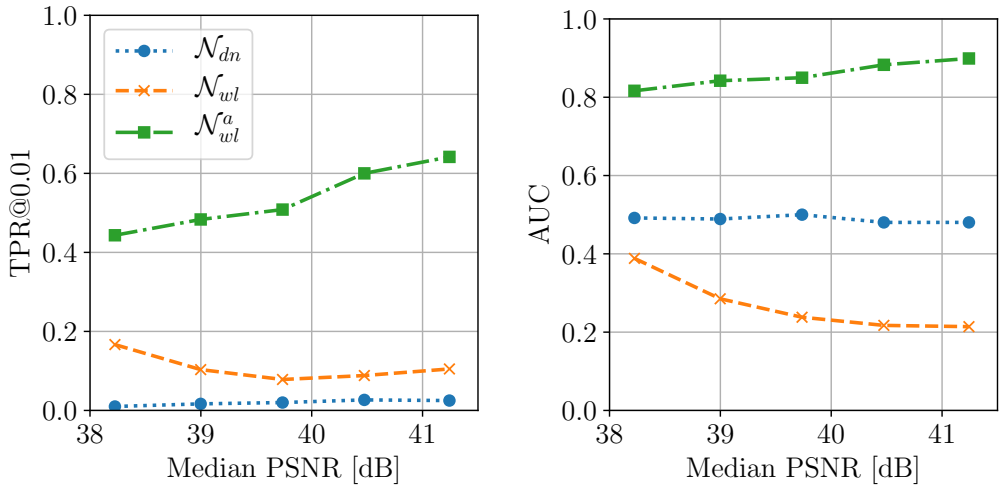


Figure 3.41: True-Positive Rate at a fixed False-Alarm probability $\alpha = 0.01$ (a) and Area Under Curve (b) when varying PSNR_{min}.

the cross-correlation value between $\boldsymbol{K}$ and $\hat{\boldsymbol{W}}$ extracted through $\mathcal{N}_{\text{dn}}$, is effectively working as expected. When noise residuals are extracted from $\hat{\boldsymbol{Y}}$ through the Wavelet-based function and the unidirectional cross-correlation test in Equation (3.5) is used ($\mathcal{N}_{\text{wl}}$), the detection performance are severely affected. However, resorting to the symmetric detector ($\mathcal{N}^{\text{a}}_{\text{wl}}$) shows that in fact the detection performances are affected, but are not as bad as when the asymmetrical detector is used.

A final result is shown in Figure 3.41, where two standard metrics in anonymization

are presented. Figure 3.41a and Figure 3.41b respectively report the True-Positive rate at a fixed False-Alarm rate of 0.01 and the Area Under Curve for several median PSNR values. Each point is obtained by setting $\mathrm{PSNR}_{\mathrm{min}}$ to $\{37, 38, 39, 40, 41\}$. The almost zero TPR@0.01 value for $\mathcal{N}_{\mathrm{dn}}$ and the almost constant 0.5 value for AUC are assessing that the anonymization cycle is working properly if the noise extraction function used in the anonymization loop is the same as the one used for analysis purposes. When a different noise extraction function is used and a forensics investigator is aware of the attack ($\mathcal{N}^{\mathrm{a}}_{\mathrm{wl}}$) the anonymization is not guaranteed anymore.

3.7 Applications

In this section we present two applications related to the use of PRNU fingerprints as features. In Section 3.7.1 we introduce a method to create robust fingerprint for smartphone-based authentication [18], merging together accelerometer, gyroscope and PRNU residual to obtain a unique identifier for the smartphone at hand. In Section 3.7.2 we present a method that exploits PRNU residuals to group together videos coming from the same camera device within a pool of semantically similar videos.

3.7.1 Robust smartphone fingerprint

Nowadays, and probably always more in the next coming years, many of our basic activities such as reading an e-mail, checking our bank account, buying on-line, etc., are performed by using a smartphone to access our personal accounts in a mobile environment. Generally, our actual degree of security is granted by the classic username and password access (something that the user knows). When a stronger level of security is required, additional instruments are usually adopted such as smart cards, USB sticks, OTP generators, in a two-factor authentication protocol [135]. Anyway, such means are not always available (must be carried around by the user at all times) or usable (they are not pluggable in a mobile device easily); so the need of a superior degree of security often conflicts with feasibility and usability. A possible solution could envisage the use of the user's own smartphone and its intrinsic characteristics as a mean to grant a safer mobile access by reducing the end-user involvement. The basic idea is to investigate and understand if it is possible to generate a specific fingerprint that allows to distinctively and reliably characterize each smartphone hardware in a unique way. As a matter of fact, modern mobile phones are equipped with several kinds of sensors such as accelerometer, gyroscope, magnetometer, camera, etc. These sensors are characterized by peculiar anomalies left onto the acquired signals due to the imperfections generated

during the manufacturing process [13]. Therefore, it is possible to measure these anomalies and exploit them as an asset for uniquely identifying each phone. The objective of this section is to present a methodology to obtain a robust smartphone fingerprint by opportunely combining different sensor fingerprints. The proposed methodology to create the smartphone fingerprint is firstly based on the individuation and definition of a set of distinctive features for each sensor; in our experiments we considered the accelerometer, the gyroscope and the camera. For the accelerometer and the gyroscope we considered two subsets of features both in the temporal and in the spectral domain, calculated onto the output data (x, y, z) acquired by each sensor [136, 137]. Concerning the camera, we computed spatial features derived from the 2D Photo Response Non-Uniformity (PRNU) noise [105, 138], extracted from the R, G, B channels. All these features, organized in a vector, constitute the fingerprint of each device. According to these fingerprints, a classifier has been trained and some experimental tests to evaluate detection performances of the method have been carried out. Also different sub-combinations of the sensors have been considered in creating the fingerprint (e.g., only the accelerometer, accelerometer and the camera, accelerometer and gyroscope, etc.) to better understand which was the impact of each sensor on distinctiveness. Moreover, to decrease computational complexity, we investigated the possibility of reducing fingerprint size through hashing operations typically used for PRNU [15, 50, 139]. Furthermore, diverse operative conditions have to be analyzed: smartphone position (handheld or posed on a table of different materials), vibration on/off, with or without a cover and so on.

3.7.1.1 Sensors overview

Most smartphone devices have built-in sensors that measure motion, orientation, and various environmental conditions. These sensors are capable of providing raw data and are useful if you want to monitor three-dimensional device movement or positioning, or you want to monitor changes in environment near a device. In general, both Android and iOS platforms, support three categories of sensors: motion sensors, environmental sensors and position sensors. The first kind of sensors measures acceleration forces and rotational forces along three axes. This category includes accelerometer, gravity sensor, gyroscope and rotational vector sensor. The environmental sensors measure various parameters, such as ambient temperature and pressure, illumination and humidity. This category includes barometer, photometer and thermometer. The last kind of sensors measure the physical position of a device. This category includes orientation sensor, GPS and magnetometer. Some of these sensors are hardware-based and some are software-based. Hardware-based sensors are physical components built into a smartphone or a tablet

Sensor	Android 4.0	Android 2.3	Android 2.2	Android 1.5
ACCELEROMETER	✓	✓	✓	✓
AMBIENT_TEMPERATURE	✓	×	×	×
GRAVITY	✓	✓	×	×
GYROSCOPE	✓	✓	×	×
LIGHT	✓	✓	✓	✓
LINEAR_ACCELERATION	✓	✓	×	×
MAGNETIC_FIELD	✓	✓	✓	✓
ORIENTATION	✓	✓	✓	✓
PRESSURE	✓	✓	×	×
PROXIMITY	✓	✓	✓	✓
RELATIVE_HUMIDITY	✓	×	×	×
ROTATION_VECTOR	✓	✓	×	×
TEMPERATURE	✓	✓	✓	✓

Figure 3.42: Sensors vs Android versions.

device. They derive their data by directly measuring specific environmental properties, such as acceleration, geomagnetic field strength, or angular change. Regarding Android-powered devices, few of them have every type of sensor. For example, most handset devices and tablets have an accelerometer and a magnetometer, but fewer devices have barometer or thermometer. On the other side, regarding operative system, all the sensors are supported from Android 4.0 (see Figure 3.42).

Accelerometer The accelerometer inside a smartphone is composed by a circuit having seismic mass (made up of silicon) that changes its position according to the orientation and it is attached to the circuit of the device. Actually an accelerometer is a circuit based on MEMS (Micro Electro Mechanical System), that measures the forces of acceleration that may be caused to gravity, to the movement or by tilting action. Such accelerations are measured in terms of g-force on the three axes (x, y, z). MEMS-based accelerometers are based on differential capacitors. Figure 3.43 shows the internal architecture of a MEMS-based accelerometer. As we can see there are several pairs of fixed electrodes and a movable seismic mass. Under no acceleration the distances d_1 and d_2 are equal and as a result the two capacitors are equal, but a change in the acceleration will cause the movable seismic mass to shift closer to one of the fixed electrodes causing a change in the generated capacitance. This difference in capacitance is detected and amplified to produce a voltage proportional to the acceleration. The minute imprecision in the electro-mechanical structure induce imperfections among the accelerometer chips.

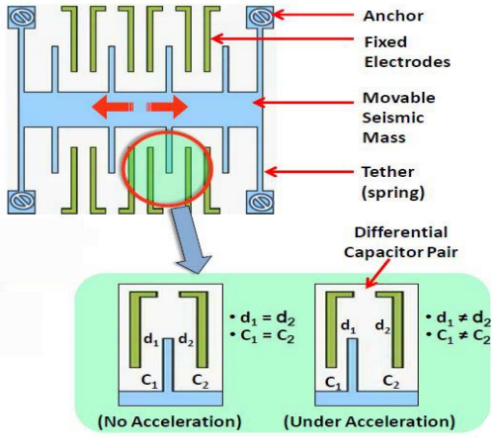


Figure 3.43: MEMS accelerometer: how it works.

Gyroscope A gyroscope is a device for measuring or maintaining orientation, based on the principle of angular momentum. Mechanically, a gyroscope is a spinning wheel or disk in which the axle is free to assume any orientation. Same as accelerometer, gyroscope returns three-dimensional values along the three axes of the device and it measures the rate of rotation (in rad/s). MEMS-based gyroscopes use the Coriolis effect to measure the angular rate. Whenever an angular velocity ω is exerted on a moving mass of weight m and velocity v, the object experiences a Coriolis force in a direction perpendicular to the rotation axis and to the velocity of the moving object. The Coriolis force is sensed by a capacitive sensing structure where a change in the vibration of the proof-mass causes a change in capacitance which is then converted into a voltage signal by the internal circuitry. The slightest imperfections in the electro-mechanical structure could introduce differences across chips.

3.7.1.2 Features vs Sensors

As mentioned in the previous section, sensors readings are affected by anomalies due to sensors imperfections. Our goal is to detect these anomalies and exploit them as an asset to understand which device generated them. To accomplish this goal, we make use of a set of features computed on signals acquired by the different sensors.

For both accelerometer and gyroscope it is possible to obtain raw values along three axes of the device at certain time. So, for a given timestamp t we have two vectors of the following form: $\boldsymbol{a}(t) = (a_x, a_y, a_z)$ and $\boldsymbol{\omega}(t) = (\omega_x, \omega_y, \omega_z)$ for the accelerometer and gyroscope respectively.

Table 3.3: List of time domain features

Feature Name	Accelerometer	Gyroscope
Mean	x	
Std-Deviation	x	x
Average Deviation	x	x
Skewness	x	x
Kurtosis	x	x
RMS amplitude	x	x
Lowest value	x	x
Highest value	x	x
ZCR		x
Non-negative count		x

As regarding accelerometer, 17 scalar features are extracted in both time and frequency domains using the MIRToolbox [140], a popular audio feature extraction library [141], starting from the two following signals:

$$T(k) = t(k+1) - t(k)$$
$$S(k) = \sqrt{a_x^2(k) + a_y^2(k) + a_z^2(k)}$$

The time domain features are calculated using $T(k)$ and $S(k)$ signals prior to interpolation, and the frequency domain features are drawn from the interpolated versions. In total a 34 features vector $\mathbf{f}_a$ is obtained to describe the accelerometer sensor. In Table 3.3 and Table 3.4 all the features taken in consideration are outlined. For a complete description of each feature please refers to the MIRToolbox guide.

Regarding the gyroscope we consider data from each axis as a separate stream in the form of $\omega_x, \omega_y, \omega_z$. For all data streams, time and frequency domain characteristics are analyzed as for the accelerometer. To summarize the characteristics of each signal, 21 features are extracted, consisting of 10 temporal and 11 spectral features (listed in Table 3.3 and Table 3.4). In total a 63 features vector $\boldsymbol{f}_g$ is used to describe the gyroscope sensor for each device.

Concerning digital cameras, we exploit a feature vector based on the PRNU. To this purpose, let $\boldsymbol{Y}$ be a 2D digital image, and $\boldsymbol{W}$ be the noise residual extracted from $\boldsymbol{Y}$, according to Equation (3.3), with the de-noising wavelet-based procedure described in [13,105]. Given a set of images $\boldsymbol{Y}_p,\ p \in \{1, ..., P\}$ from the same camera, the maximum-likelihood estimation of the PRNU for that camera $\boldsymbol{K}$ is derived from Equation (3.4). For particularly accurate PRNU estimation, the use of flatfield bright images is encouraged

Table 3.4: List of frequency domain features

Feature Name	Accelerometer	Gyroscope
Spectral Std-Dev	x	x
Spectral Centroid	x	x
Spectral Skewness	x	x
Spectral Kurtosis	x	x
Spectral Crest	x	x
Irregularity-J	x	x
Smoothness	x	x
Flatness	x	x
Roll Off	x	x
Entropy		x
Brightness		x
Roughness		x

(e.g., shots of the sky). However, $\boldsymbol{K}$ can still be estimated (with less precision) using pictures of natural scenes. In this case, a sufficient greater number of images is needed (i.e., P must be increased with respect to the flatfield case) [13].

It is well known that the correlation between the noise residual $\boldsymbol{W}$ and the PRNU $\boldsymbol{K}$ assumes high values only if $\boldsymbol{W}$ is extracted from an image coming from the camera whose PRNU is $\boldsymbol{K}$ [13, 32, 107, 142]. Therefore, a feature characterizing a digital camera could in principle be the noise component $\boldsymbol{W}$. However, for the sake of speeding up the device identification process, and motivated by state-of-the-art works on PRNU compression [50, 139], we perform an additional step. More specifically, we consider only the 512×512 pixels patch taken from the center of $\boldsymbol{W}$, and binarize it according to its sign. Formally, the feature vector used to describe a camera is $\boldsymbol{f}_c = \mathrm{sgn}(W_{512\times512})$, where $\boldsymbol{W}_{512\times512}$ is the 512×512 central portion of $\boldsymbol{W}$. Notice that, thanks to binarization, $\boldsymbol{f}_c$ can be stored using only 512×512 bits (i.e., less than 33 kBytes), which is much less than a typical image size and allows very fast feature transmission also in low bandwidth conditions.

3.7.1.3 Procedure

In order to achieve device identification, we propose a methodology based on supervised classification and the features described in the previous section (i.e., $\boldsymbol{f}_a$, $\boldsymbol{f}_g$ and $\boldsymbol{f}_c$). In the considered scenario, there are two main entities: (i) the user owning a device that wants to be identified; (ii) a trained system that analyses the data provided by the user in order to identify (or not) its device. The overall identification procedure works in three steps, as shown in Figure 3.44: (i) each new user registers into the system; (ii) the

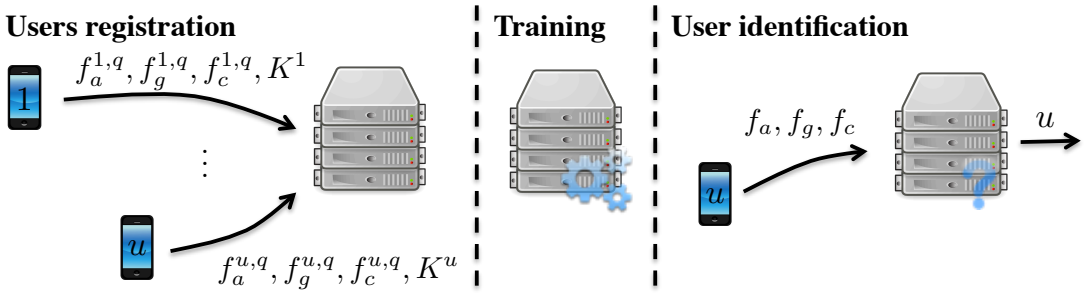


Figure 3.44: Procedure pipeline.

system is trained based on the acquired registration data; (iii) device identification can be accomplished each time a user needs it by sending a new set of features (as fingerprint) to the system.

For the registration procedure, let us consider a generic user u out of all the possible U users. First, he/she runs an application installed on its device to collect Q sensors readings from the accelerometer and the gyroscope, then finally shots $Q + P$ pictures of natural scenes (possibly neither saturated nor overly dark). From the first two sensors readings, Q different sets of feature vectors $\boldsymbol{f}_a^{u,q}$ and $\boldsymbol{f}_g^{u,q}$, $q \in \{1, ..., Q\}$ are computed. From Q images, the device computes a set of feature vectors $\boldsymbol{f}_c^{u,q}$, $q \in \{1, ..., Q\}$. From the remaining P images, the the PRNU K^u is estimated according to Equation (3.4). The user finally sends to the server all the feature sets $\boldsymbol{f}_a^{u,q}$, $\boldsymbol{f}_g^{u,q}$ and $\boldsymbol{f}_c^{u,q}$, $q \in \{1, ..., Q\}$, and the PRNU estimate $\boldsymbol{K}^u$. This procedure is followed by every user.

At this point, the system can be trained as shown in Figure 3.45. To this purpose, from each 512×512 bits training feature $\boldsymbol{f}_c^{q,u}$, a U-dimensional feature vector $\hat{\boldsymbol{f}}_c^{q,u}$ is computed, defined as

$$\hat{\boldsymbol{f}}_c^{q,u} = \left[\rho(\boldsymbol{f}_c^{q,u}, \boldsymbol{K}^1),\ \rho(\boldsymbol{f}_c^{q,u}, \boldsymbol{K}^2),\ \ldots,\ \rho(\boldsymbol{f}_c^{q,u}, \boldsymbol{K}^U)\right],$$

where ρ computes the cross-correlation as from Equation (3.5). In other words, each component of $\hat{\boldsymbol{f}}_c^{q,u}$ is the correlation value between $\boldsymbol{f}_c^{q,u}$ and one of the possible PRNU templates $\boldsymbol{K}^u$, $u \in \{1, ..., U\}$. From an intuitive point-of-view, the vector $\hat{\boldsymbol{f}}_c^{q,u}$ should point in the u-th direction, strongly indicating the correct camera. Features $\boldsymbol{f}_a^{q,u}$, $\boldsymbol{f}_g^{q,u}$ and $\hat{\boldsymbol{f}}_c^{q,u}$ are concatenated and used to train a supervised classifier (in our experiments a Bagged Decision Tree [143]). In a nutshell, a set of classifiers is trained several times over random splits of training data. Each training split leads to a different classification tree. The final classification result is obtained by majority voting over all the classification trees

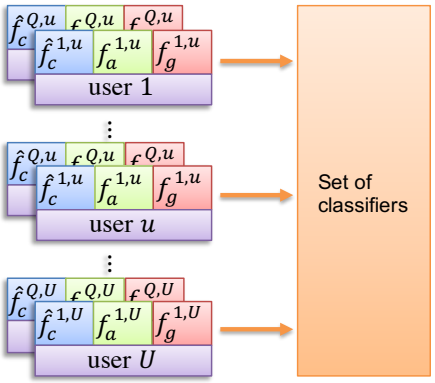


Figure 3.45: During training procedure the set of classifiers is fed with $Q \times U$ feature vectors with the corresponding labels. Each feature vector is obtained by concatenating $\hat{\boldsymbol{f}}_c^{q,u}$, $\boldsymbol{f}_a^{q,u}$, $\boldsymbol{f}_g^{q,u}$.

Once the system has been properly trained, each time a user needs to be identified, he/she can simply collect a few seconds sensors reading and shoot a picture. The user's device computes the tuple of features $\boldsymbol{f}_a$, $\boldsymbol{f}_g$ and $\boldsymbol{f}_c$ and sends them to the server. As per training, feature $\boldsymbol{f}_c$ is converted into $\hat{\boldsymbol{f}}_c$, and classification is performed using the concatenation of $\boldsymbol{f}_a$, $\boldsymbol{f}_g$ and $\hat{\boldsymbol{f}}_c$ as input for the classifier.

3.7.1.4 Experimental Results

Different experimental tests have been carried out to verify the effectiveness of the proposed methodology and some of them are presented hereafter in this section. In particular the dataset of the considered smartphones is described (only Android platform has been analyzed) together with the way sensor acquisitions are performed, then we report the diverse test scenarios that have been investigated. Finally, we presents the achieved results in the various circumstances.

Test setup Experimental tests have been carried out on 10 different smartphones that are listed hereafter in Table 3.5. It is worthy to highlight that one half are the same model and this has been chosen in order to investigate which was the actual capacity to distinguish also within devices of the brand and model. The acquisitions from the sensors have been done by means of a specific mobile application, named *SensorData* which is able to interact with the smartphone sensors and get their output signal. Both for the accelerometer and for the gyroscope 20 acquisitions ($Q = 10$ for training plus 10 for testing), of 2 seconds each, have been taken. Because of the different characteristics of every smartphones, the number of samples within each acquisition is diverse, so each

Table 3.5: List of smartphones

Device	Amount
LG Nexus 5	5
Samsung S2plus	1
Samsung S3	2
Samsung S4	1
Motorola	1

sequence has been resampled by using spline interpolation to compute spectral features. For what concerns digital images, they have been taken at the default resolution and settings of the device, which is (in pixels): 2448×3264 (Nexus5, S2plus and S3), 1836×3264 (Motorola) and 2322×4128 (S4). For each camera, $P = 10$ images have been used for PRNU estimation, $Q = 10$ for training and other 10 for testing. Notice that only the central 512×512 pixels portion of each image has been used, thus no image resampling is needed even if different devices have different camera resolutions.

Test scenarios and evaluation metric Different test scenarios have been envisaged in order to understand which could be the operative circumstances that could affect performances. There are, in fact, many aspects that can influence the acquisition phase both during training and testing steps: first of all, the smartphone's position (leaning on a table, hand-held by a still or slightly moving user, etc.), secondly, the usage conditions (characteristics of the table surface, presence or not of a telephone cover made of diverse materials, audio/vibration stimulation, running processes on the operating system, etc.) and so on. In the next subsection, some of the main achieved results are presented; in particular, two basic cases have been taken into account: when the acquisition takes place in a more controlled environment with the smartphone on a flat wooden surface and when it is hand-held (without any cover in any case). Tests have been carried out in order to understand how different conditions can impact on training with respect of testing and viceversa. The issue of using or not a vibrating impulse has been considered when acquiring only from the accelerometer, but no requirements have been imposed on the running processes at all. Classification has been done by resorting at a Bagged Decision Tree approach and experimental tests has been carried out both for each sensor separately and also in combination. The obtain results have been evaluated in terms of *F-score (F)* which is defined as in Equation (3.28):

$$F = (2 \cdot Pr \cdot Re)/(Pr + Re) \tag{3.28}$$

Table 3.6: Test configurations

	Training set		**Test set**	
	Position	*Vibration*	*Position*	*Vibration*
Configuration1	Table	ON	Table	ON
Configuration2	Table	OFF	Table	OFF
Configuration3	Hand-held	ON	Hand-held	ON
Configuration4	Table	ON	Hand-held	ON
Configuration5	Hand-held	ON	Table	ON

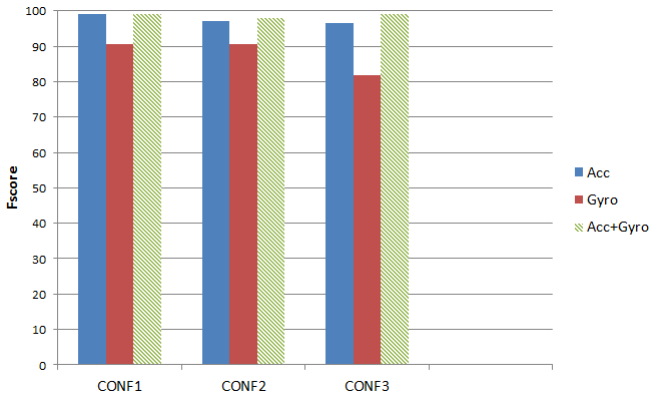


Figure 3.46: F-score in percentage for the scenarios where training and testing are aligned.

where $Pr = TP/(TP + FP)$ and $Re = TP/(TP + FN)$ stands for *Precision* and *Recall*. The overall *F-score* is the average of *F-score* computed on each class.

Results This Section presents experimental results with reference to some test configurations exploited in Table 3.6. When the parameter "Vibration" is set up ON, it is intended that the acquisition for the accelerometer sensor has been done when there was a stimulation generated by the vibration motor.

It is interesting to underline that the two last test configurations conceive that training and testing conditions are not aligned. This circumstance has revealed as being more challenging, as expected, but it represents an actual operative scenario where there is no control over the end user activities. In Figure 3.46 the F-score values for the first three test configurations are depicted. It can be seen, as general, that the two sensors, accelerometer and gyroscope, are both able, by themselves, to provide reliable results in terms of device distinctiveness; however, when both are used together performances are improved with a F-score that tends to achieve values around 100%.

In particular, for the case of *Configuration3* (Hand-held) which is more challenging, it

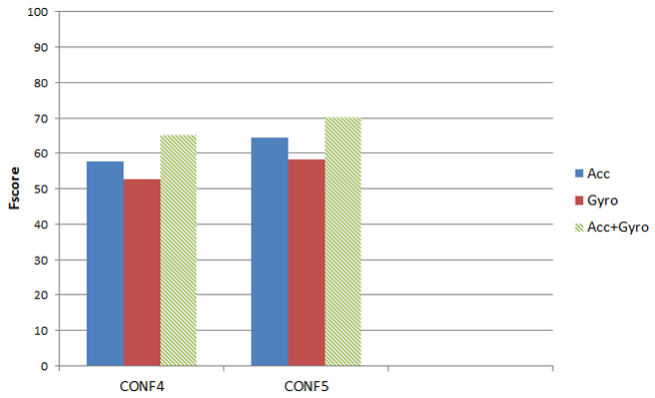


Figure 3.47: F-score in percentage for the scenarios where training and testing are not aligned.

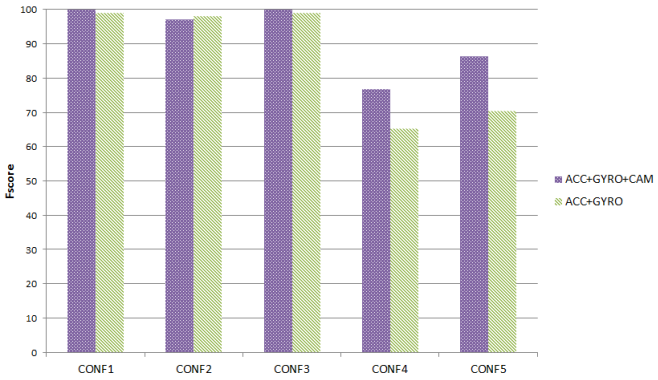


Figure 3.48: F-score in percentage for all the scenarios when the camera sensor is added.

is interesting to notice that a high value of F-score is achieved similarly to what happened for the other two configurations in which the smartphone is leaning on the table.

On the contrary, in Figure 3.47 the results obtained for *Configuration4* and *Configuration5* (e.g. training and testing misaligned) are presented. It is immediate to comprehend that performances are drastically worsened and not even the approach to join both sensors, though providing some benefits, does not succeed into granting satisfactory F-score values: 70% is achieved for *Configuration5* at most.

In Figure 3.48, the results obtained when also the camera sensor is taken into account are shown. It can be evidenced that the first and the third configurations, that already presented very good performances, tends to reach 100% while the second one remains almost unaltered, even a bit lower, and this could be due to the fact that not using the vibration for the accelerometer impacts on performances as a whole.

It is appreciable that for *Configuration4* and *Configuration5* results are strongly im-

proved and, for instance, a F-score of about 85% is reached in the last case.

In the light of these results, it is worth making a specific comment on the used camera fingerprint. As a matter of fact, PRNU is known to be very robust and reliable, ensuring very high accuracy in camera identification (often higher than 90%), especially when used for high-end devices. However, in our scenario, we must consider a series of constraints: (i) the user cannot be asked to shoot too many pictures for system training; (ii) we typically have no control on the kind of pictures the user sends; (iii) the generated feature vector must be small enough to enable transmission also in low-bandwidth cases. Therefore, we work in a very disadvantaged scenario: (i) PRNU is estimated using only a few (i.e., $P = 10$) images; (ii) these images represent natural scenes and are not flatfield; (iii) the correlation procedure is performed using a strongly quantised (i.e., binarisation according to the sign) image noise $\boldsymbol{W}$; (iv) smartphones cameras are strongly affected also by other noise component with respect to high-end devices. This is the main reason we cannot expect to reach an even higher accuracy using camera fingerprint in this scenario.

In Figure 3.49 the test *Configuration5* when also the sensor camera is taken into account is analyzed in detail. The confusion matrix structured onto the ten target/output classes is presented; over the diagonal there are the correct classification (green blocks) while all the other are wrong (red blocks). It is interesting to highlight that most of the performance decrement is given by the classes numbered 1 and 2 (top-left of the matrix) otherwise performances would be around a F-score of 95%.

3.7.2 Near-duplicate video detection

The forensic community has developed a wide set of algorithms to blindly reconstruct the history of a video sequence based on single video analysis [8, 144–146]. Additionally, in the last few years, researchers have also shown the possibility of performing interesting forensic analysis by jointly studying and processing multiple near-duplicate (ND) video objects, i.e., sequences obtained applying a set of content preserving transformations to the same original content [147–149]. As an example, when multiple ND copies of the same video of sensitive nature are diffused online, it is possible to help investigators pointing out which user first posted the original content. This can be done by reconstructing the video phylogeny tree, i.e., a directed graph summarizing the parental relationships among ND videos [150, 151]. Additionally, exploiting ND analysis, it is possible to help a forensic investigator to increase the media coverage about an important event of interest (e.g., to gain more information about a crime scene [152]).

A fundamental step in this kind of algorithms is the accurate collection of ND video sequences to analyze. To this purpose, many algorithms robust against video transforma-

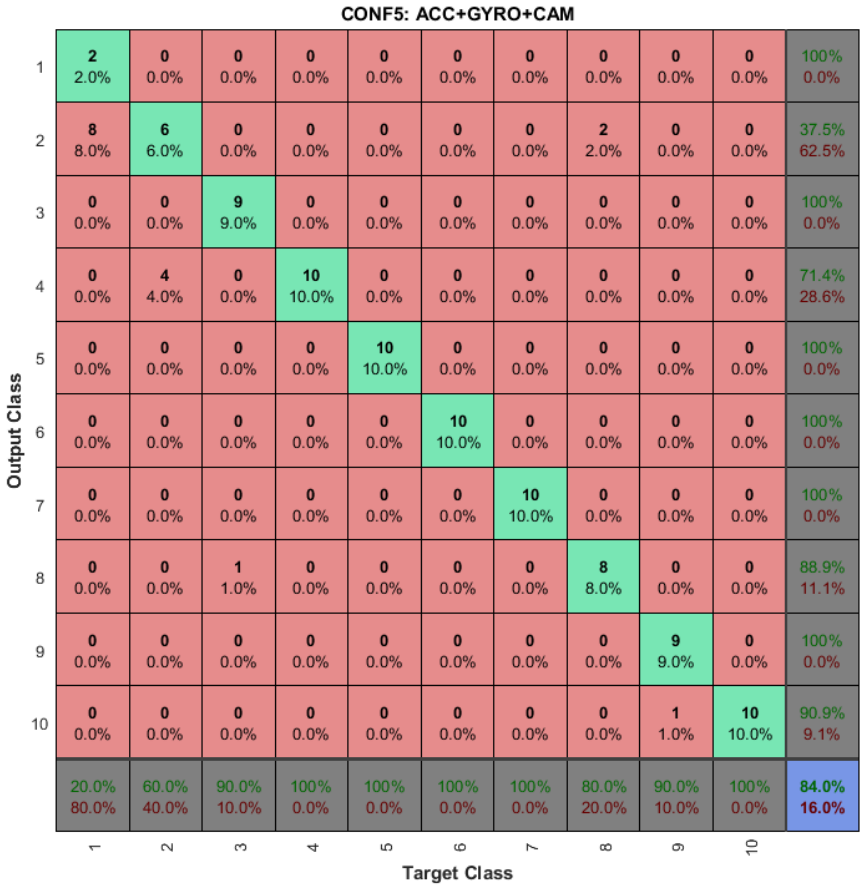


Figure 3.49: Case Configuration5 (considering also sensor camera): confusion matrix.

tions have been proposed in the video search and retrieval literature [153, 154]. However, in a forensic scenario, it is not uncommon that analysts investigate public events of interest (e.g., public speeches, criminal attacks, etc.). These are often simultaneously documented by many people using their own capturing devices (e.g., smartphones) from different viewpoints, thus giving rise to semantically similar (SSI) videos in addition to NDs (see Figure 3.50). Algorithms tailored to ND video detection that do not take this possibility into account may incorrectly recognize as NDs also SSI videos acquired by different users from particularly close points of view. As an example, the authors of [155] showed that the hashing-based algorithm for ND detection discussed in [152] can be used to detect also SSI video sequences. This is undesirable for video phylogeny algorithms that strictly process ND videos and do not deal with SSI ones [149–151].

In order to solve the aforementioned problem, in this section we propose a ND video detection algorithm, which is robust against SSI sequences even if coming from very close

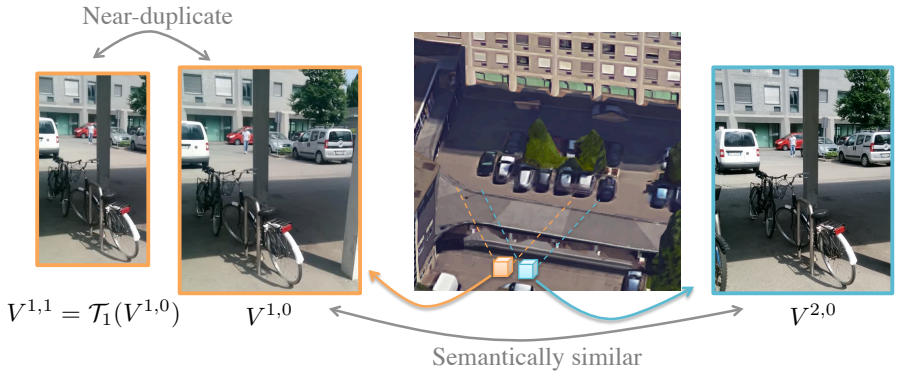


Figure 3.50: Example of ND and SSI videos. A scene is captured simultaneously with two devices, originating $V^{1,0}$ and $V^{2,0}$. Video $V^{1,0}$ is further edited to obtain a ND video $V^{1,1}$. In this scenario it is difficult to distinguish videos coming from different devices.

viewpoints. Given a pool of videos under analysis, the goal is to blindly cluster groups of NDs, not containing spurious SSI sequences. The proposed method is based on two founding concepts: (i) robust video hashing [151, 154] to perform a first rough and fast screening; (ii) sensor noise traces left on video sequences by the capturing device [156,157], for detection refinement.

The rationale behind the proposed method is that ND videos are all generated from the same original sequence, thus they come from the same acquisition device. Conversely, SSI videos recorded simultaneously are by definition acquired from different cameras. Therefore, after we group SSI videos based on semantical content using robust hashing techniques, we can resort to traces characterizing different devices to recognize ND families. Specifically, as it is well known that each device leaves on captured videos peculiar footprints that can be exposed by denoising operations [105, 156, 157], we rely on this concept for the refinement step.

The validation campaign is performed on a set of more than 12 000 videos among ND and SSI ones coming from seven different devices. Experiments show that the proposed approach is capable of reaching more than 98% of accuracy in ND detection, thus greatly improving over the baseline ND detection method [152] based only on robust hashing. We also tested the proposed algorithm on a set of videos gathered online.

3.7.2.1 Problem Formulation

In everyday scenarios, events of interest (e.g., public speeches, criminal attacks, but also concerts and sport events) are often documented by different users that simultaneously capture the scene with their own devices from multiple viewpoints. In this context, let

us define $\{V^{p,0}\}_{p\in[1,P]}$ a set of P original videos, each one acquired by the p-th device (i.e., from the p-th point of view).

If these videos are shared, other users can edit and re-distribute their own copies, thus creating other J_p versions $V^{p,j} = \mathcal{T}_j(V^{p,0})$, $j \in [1, J_p]$ of each original content $V^{p,0}$, where $\mathcal{T}_j$ is any content preserving transformation (e.g., colour enhancement, logo insertion, resizing, frame cropping, or combinations of them). As an example, this happens every time newscasts broadcast the same interview at different resolutions with different overlay text and superimposed logos.

In this scenario, all videos $V^{p,j},\ p \in [1, P],\ j \in [0, J_p]$ are defined *semantically similar* (SSI) sequences [155], since they all capture the same semantic information about a scene or an event. In particular, videos $V^{p,j},\ j \in [0, J_p]$ for a fixed p value are denoted as strictly *near-duplicate* (ND) sequences [149, 150, 152]. Figure 3.50 shows an example of the ND and SSI videos generative process.

Given a generic set of video sequences, ND video detection refers to the problem of correctly clustering separate groups of strictly ND videos. Solving this problem is paramount for video phylogeny tools designed to jointly analyze multiple versions of the same video object [149, 150], as they work under the hypothesis that only NDs are present within the analysis pool. For this reason, some effective ND detection solutions have been proposed in the literature [153, 154]. However, when the set of considered video sequences contains SSI videos shot from very close viewpoints (as those shown in Figure 3.50), ND detection problem turns out to be more challenging. Indeed, solutions as the one proposed in [152] tend to cluster SSI videos as if they were NDs [155].

In this work we propose an algorithm for ND video detection in this challenging scenario. Specifically, our goal is to separate actual ND sequences from SSI sequences captured from different viewpoints, even if very close to each other. Formally, given a set of videos $\{V^{p,j}\}_{p\in[1,P],\, j\in[0,J_p]}$, we aim at individuating: (i) the number P of ND clusters; (ii) the composition of each cluster $C_p = \{V^{p,j}\}_{j\in[0,J_p]}$, $p \in [1, P]$, each one containing only ND videos generated from the original sequence $V^{p,0}$.

3.7.2.2 Near Duplicate Video Detection

In this section we present the proposed algorithm for ND video detection starting from a generic set of SSI videos relative to the same event $\mathcal{V} = \{V^{p,j}\}_{p\in[1,P],\, j\in[0,J_p]}$. For notation simplicity, from now on, a generic video in $\mathcal{V}$ will be denoted as V_i, univocally mapping the pair of indexes (p, j) into a single index i.

The proposed algorithm works by comparing pairs of videos (V_{i_1}, V_{i_2}) in $\mathcal{V}$ separately in three stages, as shown in Figure 3.51. When all video pairs in $\mathcal{V}$ have been compared

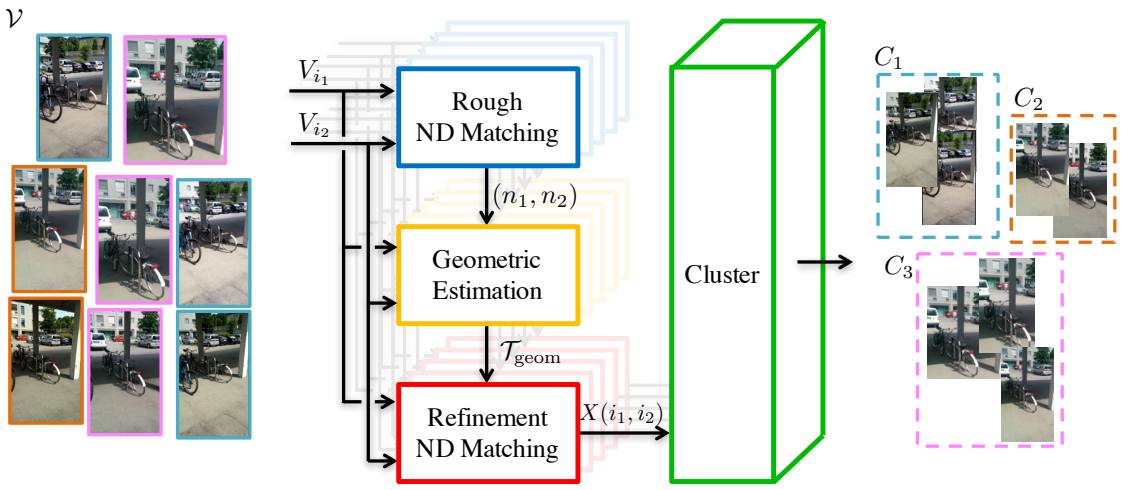


Figure 3.51: Pipeline of the proposed method. A pool of SSI videos is analyzed, and separate clusters of NDs are identified.

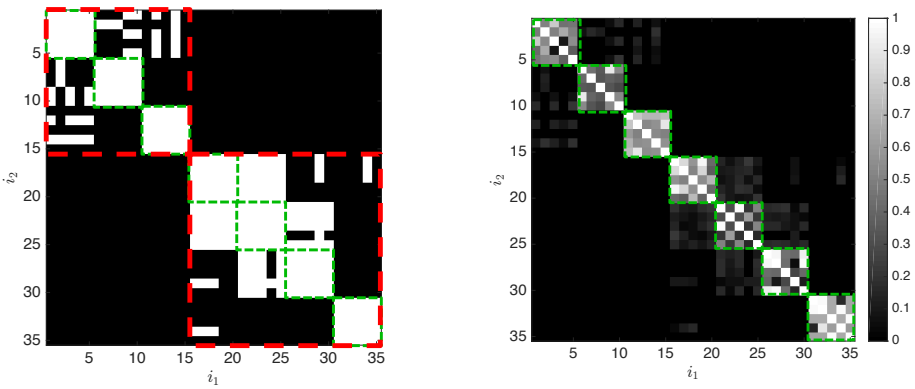
to estimate how likely they are ND, a clustering algorithm is applied to group separate sets of ND video sequences. In the following a detailed description of each step.

Rough ND matching The first step of the algorithm consists in performing a rough and fast ND detection based on the robust hashing method proposed in [151] to determine whether sequences V_{i_1} and V_{i_2} are ND candidates. Indeed, if the two videos happen to be ND or SSI depicting the same time instant from closed viewpoints, the algorithm in [154] returns the sets of corresponding frames of V_{i_1} and V_{i_2} that are temporally synchronized. Depending on the output of [151], we fill a logical near-duplicate relationship matrix defined as

$$M_{i_1,i_2} = \begin{cases} 1, & \text{if synchronized frames are found,} \\ 0, & \text{otherwise.} \end{cases} \tag{3.29}$$

If $M_{i_1,i_2} = 1$, we detect videos V_{i_1} and V_{i_2} as candidate ND and apply the following refinement steps of the algorithm. If $M_{i_1,i_2} = 0$, we proceed analyzing the next video pair.

Note that M can be interpreted as a binary adjacency matrix representing a undirected graph, whose nodes represent videos, and video pairs (V_{i_1}, V_{i_2}) for which $M_{i_1,i_2} = 1$ are linked by an edge. As a matter of fact, the ND detection algorithm proposed in [152] is equivalent to running a Depth-First Search (DFS) algorithm on M to find the connected components of the graph. However, in presence of SSI videos, this simple strategy is not



(a) M_{i_1,i_2} (b) M_{i_1,i_2}

Figure 3.52: Matrices $\boldsymbol{M}$ and $\boldsymbol{X}$ computed on $\mathcal{V}$ composed by 35 videos split into 7 SSI clusters of 5 ND sequences each depicting the scene represented in Figure 3.50. Only clustering on $\boldsymbol{X}$ leads to the correct results (in green), whereas considering $\boldsymbol{M}$, sequences are under-clustered (in red).

sufficient. An example of $\boldsymbol{M}$ computed on a set of 35 videos split into 7 SSI groups of 5 NDs each, is shown in Figure 3.52(a). Correct clusters of NDs are highlighted in green, whereas connected components identified by DFS are reported in red. This motivates our further analysis.

Geometric estimation If V_{i_1} and V_{i_2} are detected as ND candidate (i.e., $M_{i_1,i_2} = 1$), we need to estimate the geometric transformations $\mathcal{T}_{\text{geom}}$ that maps V_{i_1} into V_{i_2} in terms of resize and crop, before further proceeding with noise analysis. To do so, we select a pair of matching frames $(V_{i_1}(s_1), V_{i_2}(s_2))$ returned by [151], and estimate the homography and crop transformation $\mathcal{T}_{\text{geom}}$ between them by matching points of interest using Speeded-Up Robust Features (SURF) [158] and applying RANSAC [159] (see Figure 3.53).

Note that we estimate $\mathcal{T}_{\text{geom}}$ only once for each video pair, assuming that the geometric transformation mapping V_{i_1} into V_{i_2} is exactly the same for all the sequences' frames, which is typical for ND videos. However, in order to increase algorithm's robustness, it is possible to compute a different transformation $\mathcal{T}_{\text{geom}}$ for each pair of frames returned by [151].

Refinement ND matching Given the video pair V_{i_1} and V_{i_2} and the geometrical transformation $\mathcal{T}_{\text{geom}}$, the rationale of the refinement step is to leverage camera fingerprint to assess whether the two potential ND videos have been actually acquired by the same device (i.e., they really are NDs). To this purpose, we estimate sequences' fingerprints $\hat{\boldsymbol{K}}_{i_1}$ and $\hat{\boldsymbol{K}}_{i_2}$ by aggregating noises extracted from a set of S frames as typically done

in camera fingerprinting works [105, 156]. This can be done also in the ND scenario as camera traces are not deleted by cropping or resizing [112, 157].

Formally, the noise residual of each frame $\boldsymbol{V}_s$ of a video is defined as $\boldsymbol{W}_s = \boldsymbol{V}_s - \mathcal{D}(\boldsymbol{V}_s)$, where $\mathcal{D}$ is a denoising operator. Given a set of frames $\boldsymbol{V}_s,\ s \in [1, S]$ from the same camera, the fingerprint can be computed as

$$\hat{\boldsymbol{K}} = \frac{\sum_{s=1}^{S} \boldsymbol{W}_s \circ \boldsymbol{V}_s}{\sum_{s=1}^{S} \boldsymbol{V}_s^2}. \tag{3.30}$$

Note that, as we are considering ND videos, they may have different resolutions due to the processing operations they underwent, e.g, resize and cropping. Therefore, before comparing two different fingerprints $\hat{\boldsymbol{K}}_{i_1}$ and $\hat{\boldsymbol{K}}_{i_2}$ it is necessary to geometrically register them. To this purpose, we apply to $\hat{\boldsymbol{K}}_{i_1}$ the estimated transformation $\mathcal{T}_{\text{geom}}$, thus obtaining $\hat{\boldsymbol{K}}_{i_1 \to i_2} = \mathcal{T}_{\text{geom}}(\hat{\boldsymbol{K}}_{i_1})$, i.e, the warped version of $\hat{\boldsymbol{K}}_{i_1}$ into $\hat{\boldsymbol{K}}_{i_2}$. We compare the registered noise fingerprints by means of the normalized cross-correlation (NCC) $\rho(\hat{\boldsymbol{K}}_{i_1 \to i_2}, \hat{\boldsymbol{K}}_{i_2})$, and store this value in a cross-correlation matrix $\boldsymbol{X}$ defined as

$$X_{i_1,i_2} = \rho(\hat{\boldsymbol{K}}_{i_1 \to i_2}, \hat{\boldsymbol{K}}_{i_2}). \tag{3.31}$$

The higher the X_{i_1,i_2} value, the higher the probability of V_{i_1} and V_{i_2} coming from the same device (i.e., are NDs). Conversely, low X_{i_1,i_2} values are expected for sequences coming from different devices (i.e., SSI videos). An example is reported in Figure 3.52(b).

Note that this registration step is paramount for two reasons. First, it compensates for ND geometrical transformations allowing us to correctly match NDs as explained. Second, if we compare two videos from the same camera that are not NDs (i.e., they are not obtained from the same original video) it allows us to correctly detect them as non NDs. Indeed, in this scenario, the estimated $\mathcal{T}_{\text{geom}}$ would be a meaningless transformation leading to fingerprint desynchronization. Therefore, noise traces would not match, and we would not incorrectly detect as NDs videos that actually are not.

Finally, it is also important to notice that we do not really need to estimate a clean camera fingerprint for our goal. Indeed, if some scene content leaks into the estimated fingerprint, it helps us in matching ND videos through correlation. Therefore, even if in principle many video frames are needed to correctly estimate the camera noise for attribution problems, in our scenario we can simply exploit a reduced set of frames.

Clustering Once the comparison between all pairs of candidate ND videos are carried out, we run a clustering algorithm on rows of matrix $\boldsymbol{X}$. This step returns the separate

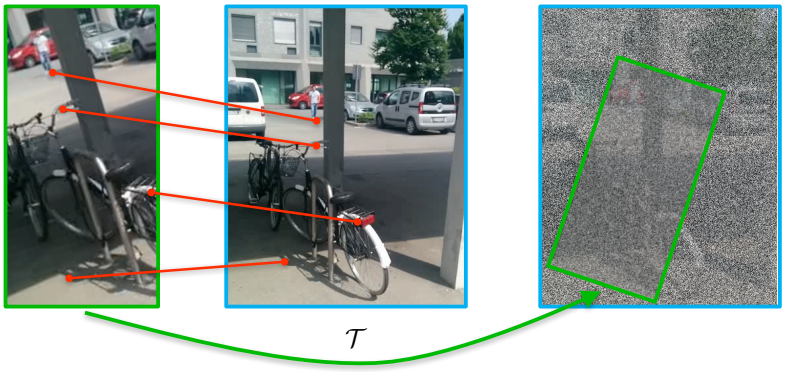


Figure 3.53: Keypoint detection and matching for $\mathcal{T}_{\text{geom}}$ estimation between two ND frames.

sets of ND videos C_p, $p \in [1, P]$.

3.7.2.3 Results

In this section we first describe the datasets used for the experimental campaign, then we report the achieved results.

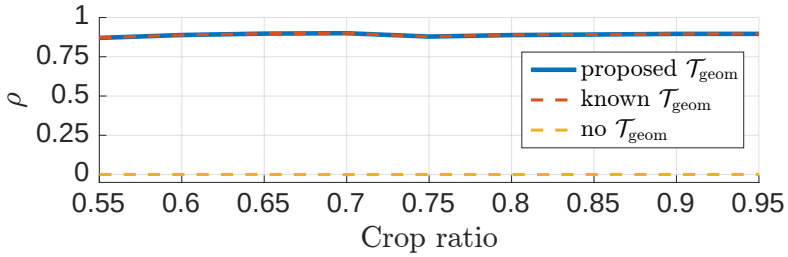
Datasets We acquired a set of SSI sequences simulating a scenario in which multiple users simultaneously take videos of the same scene. Then, NDs were generated from each SSI video through editing transformations. We opted for this strategy instead of using any common multiview video dataset available in the literature, as these datasets are usually acquired with high-end devices and cameras movements are constrained, thus leading to less realistic results.

For the SSI generation process, 7 users with 7 handheld devices acquired 9 scenes (between 15 s and 40 s each) of different nature (e.g., indoor, outdoor, moving objects, buildings, people, repeating patterns, etc.) from very close viewpoints. To be more realistic, users were free to pan or slightly rotate, given that each camera was centered on the scene of interest. Videos were not temporally synchronized. This process led to 9 families of 7 original SSI videos each, which were resized to a common 640×360 resolution. Starting from these 63 original sequences, we built different datasets to test the proposed algorithm (see Table 3.7).

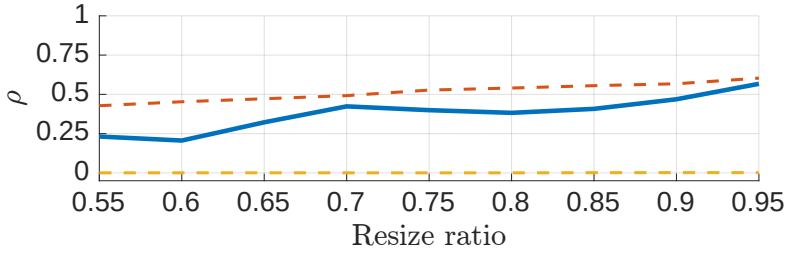
To evaluate the effect of $\mathcal{T}_{\text{geom}}$ estimation and its application to PRNUs, we built $\mathcal{V}_{\text{res}}$ and $\mathcal{V}_{\text{crop}}$ composed by 1 386 videos. Each one contains 63 sets (i.e., 9 scenes for 7 devices) of 11 sequences: an original video plus 10 ND copies obtained through resizing ($\mathcal{V}_{\text{res}}$) or cropping ($\mathcal{V}_{\text{crop}}$) to a resolution that ranges from 95% to 55% of the original one.

Table 3.7: Each dataset is composed by different video sets, characterized by different ND clusters and transformations.

Dataset	Sets (scene x realiz.)	Videos	Clusters	Transf	Tot
$\mathcal{V}_{\text{res}}$	63 (9x7)	11	1	resize	693
$\mathcal{V}_{\text{crop}}$	63 (9x7)	11	1	crop	693
$\mathcal{V}^2_{\text{ND}}$	90 (9x10)	10	2	any	900
$\mathcal{V}^3_{\text{ND}}$	90 (9x10)	15	3	any	1350
$\mathcal{V}^4_{\text{ND}}$	90 (9x10)	20	4	any	1800
$\mathcal{V}^5_{\text{ND}}$	90 (9x10)	25	5	any	2250
$\mathcal{V}^6_{\text{ND}}$	90 (9x10)	30	6	any	2700
$\mathcal{V}^7_{\text{ND}}$	90 (9x10)	35	7	any	3150



(a) $\mathcal{V}_{\text{crop}}$



(b) $\mathcal{V}_{\text{res}}$

Figure 3.54: NCC obtained on ND videos in $\mathcal{V}_{\text{crop}}$ and $\mathcal{V}_{\text{res}}$ at different resolutions applying the proposed pipeline. If geometric transformations are not compensated (yellow line), NCC tends to zero.

To evaluate the overall proposed pipeline, we generated six additional datasets $\mathcal{V}^p_{\text{ND}}$, $p \in [2, 7]$ for a total amount of 12 150 videos. Each $\mathcal{V}^p_{\text{ND}}$ contains 90 sets (i.e., 9 scenes for 10 random ND realizations) of p clusters of 5 ND videos. For ND generation, we considered the following transformations [151]: contrast enhancement, brightness adjustment, spatial cropping and resizing, in any combination. Each transformation was followed by compression with a random codec (MPEG-2, MPEG-4 Part 4, H.264/AVC), group of picture (1 to 15) and quality parameter (1 to 10).

Geometric transformation First, we evaluated the performance of fingerprint registration. We know from [112, 157] that fingerprints survives ND editing steps, but we need to validate the reliability of using $\mathcal{T}_{\text{geom}}$ to compensate for fingerprints geometric transformations.

To this purpose, for each set of videos in $\mathcal{V}_{\text{res}}$ and $\mathcal{V}_{\text{crop}}$, we analyzed the 10 video pairs $(V_i,\ V_0)$, $i \in [1, 10]$, where V_0 is the original SSI of the set and V_i is a resized or cropped version. For each pair, we estimated the fingerprints $\hat{\boldsymbol{K}}_i,\ \hat{\boldsymbol{K}}_0$ aggregating the residuals extracted from the first $S = 20$ frames of each video. We computed both $\hat{\boldsymbol{K}}_{i\to 0}$ and $\hat{\boldsymbol{K}}_{0\to i}$ by means of $\mathcal{T}_{\text{geom}}$ estimated from V_i and V_0. Evaluation is carried out by computing normalized cross-correlations (NCCs) between registered fingerprints. The higher the NCC, the better the geometrical registration.

Figure 3.54a and Figure 3.54b show NCC values at different resolutions obtained on $\mathcal{V}_{\text{crop}}$ and $\mathcal{V}_{\text{res}}$, respectively. Solid blue lines represent the average between NCC values $\rho(\hat{\boldsymbol{K}}_{i\to 0}, \hat{\boldsymbol{K}}_0)$ and $\rho(\hat{\boldsymbol{K}}_{0\to i}, \hat{\boldsymbol{K}}_i)$ (i.e. the proposed approach). Yellow dashed lines represent NCC values $\rho(\hat{\boldsymbol{K}}_{i_1}, \hat{\boldsymbol{K}}_{i_2})$ obtained without spatially synchronizing fingerprints. Dashed orange lines represent the NCC upper bounds obtained when $\mathcal{T}_{\text{geom}}$ is known a priori. These results confirm the importance of estimating $\mathcal{T}_{\text{geom}}$. Indeed, if $\mathcal{T}_{\text{geom}}$ is not applied (yellow line), NCCs tend to zero (i.e., we cannot recognize videos from the same device). Conversely, the proposed approach (blue line) enables to achieve high NCC values that allow to detect ND sequences.

Overall pipeline We applied the proposed pipeline to every set of videos in datasets $\mathcal{V}^p_{\text{ND}},\ p \in [2, 7]$. Each set is composed by p clusters of ND videos. The goal is to detect: (i) the number of clusters; (ii) which videos belong to each cluster (i.e., ND videos). For both tasks, we tested two different clustering approaches on X: (i) DFS$(\boldsymbol{X}, \Gamma)$ denotes the use of Depth First Search algorithm on matrix $\boldsymbol{X}$, binarized according to a threshold $\Gamma = 0.06$ (learned on a small training set); (ii) Hier$(\boldsymbol{X}, \Gamma)$ denotes hierarchical clustering on $\boldsymbol{X}$ using correlation distance and euclidean linkage, where $\Gamma = 1$ represents the cutoff threshold (learned on a small training set). The first approach is based on the knowledge that $\boldsymbol{X}$ is composed by NCC values of fingerprints. Therefore it links in the same cluster videos whose NCC is grater than Γ. The second approach makes use of rows of $\boldsymbol{X}$ as generic features, applying the hard Γ thresholding only at the very end.

Figure 3.55 shows the accuracy in correctly detecting the number of ND clusters within each video set. Accuracy is always greater than 80%. If the number of clusters to be detected is small (i.e., 2) the DFS$(\boldsymbol{X}, \Gamma)$ strategy tends to better results. Conversely, the Hier$(\boldsymbol{X}, \Gamma)$ approach has a better accuracy (90%) when the number of ND clusters is greater than 4. DFS applied on $\boldsymbol{M}$ [152] aggregated more ND clusters together, thus it was never able to correctly detect the number of clusters.

Evaluation of clustering approaches is carried out by measuring how well pairs of ND are assigned to the same cluster. Specifically, we used the *Rand index* (RI) [160] (the

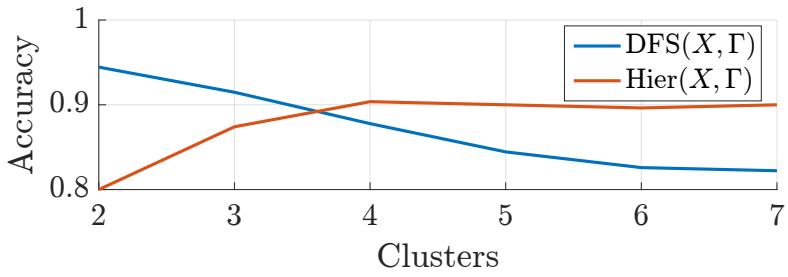
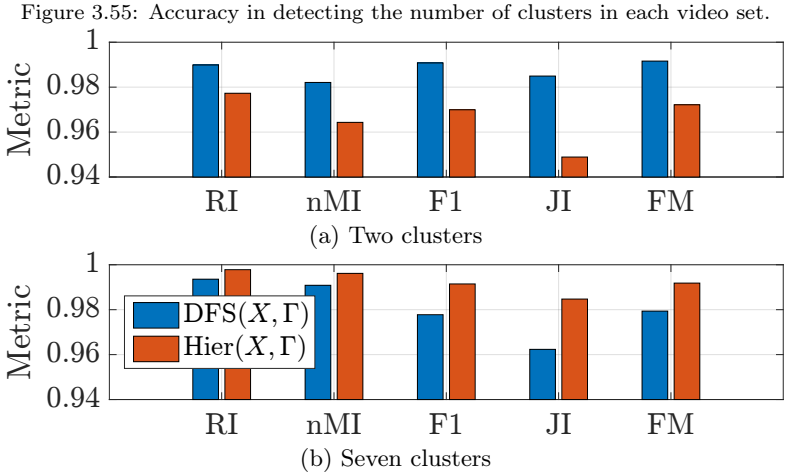


Figure 3.55: Accuracy in detecting the number of clusters in each video set.



(a) Two clusters

(b) Seven clusters

Figure 3.56: Clustering metrics obtained with the proposed techniques on datasets that contain two (a) and seven (b) ND clusters.

percentage of data pairs that are correctly clustered), the *Jaccard index* [161] (JI) (similar to RI, not considering pairs of elements that are in different clusters), the *F-measure* (F1) [162] (the harmonic mean of precision and recall), the *Fowlkes-Mallows index* (FM) [163] (the geometric mean of precision and recall), and the *normalized mutual information* measure (nMI) [164](residual entropy within clusters). All clustering measures yields values between 0 (worst result) and 1 (best result).

Figure 3.56 shows the aforementioned metrics on $\mathcal{V}^2_{\text{ND}}$ (a) and $\mathcal{V}^7_{\text{ND}}$ (b). In the first case, only two ND clusters are present, and DFS$(\boldsymbol{X},\Gamma)$ shows more promising results. Conversely, the second scenario validates the use of Hier$(\boldsymbol{X},\Gamma)$ when more clusters of NDs are present within the analysis pool.

3.8 Conclusions

In this chapter we proposed a novel projection framework for PRNU fingerprint and residuals compression based on SNR near maximization of the alternative hypothesis in a cross-correlation statistical test. We derived systematic design conditions for the

projection matrix, taking into account local interpolation effects on PRNU due to demosaicing and JPEG compression. Two projection design strategies have been introduced and tested on real-world datasets, in comparison with state-of-the-art PRNU compression methods. The obtained experimental results confirm that proposed projection methods perform at par or better with state-of-the-art Gaussian Random Projection, with a significant reduction in terms of computational complexity.

We also presented a compression pipeline for PRNU fingerprints and residuals based on decimation, Random Projections and dead-zone quantization. At first we observed that JPEG compression strongly attenuates high frequency components of the PRNU, basically zeroing the usefulness of such frequencies in terms of cross-correlation. Exploiting this phenomenon, we decimate the extracted PRNU fingerprint and residuals before passing to Random Projections. Finally we are able to further reduce the bitrate by adopting a dead-zone quantization scheme, that fuses the advantages of fingerprint binarization and digest compression methods. On the Dresden Image Dataset, the proposed pipeline accounts for more than 65% bitrate reduction with respect to basic Random Projections applied to the whole fingerprint or residual, both in terms of query and joint compression, with an overall 75% complexity reduction.

As for PRNU-based antiforensics, we proposed a pipeline for no-reference image anonymization against PRNU-based detectors. This approach is based on image inpainting to reconstruct image pixels, and edge processing to increase image visual quality. We tested different inpainting strategies, showing that it is possible to attenuate PRNU traces even exploiting simple inpainting solutions. Considering that results are competitive with state-of-the-art blind PRNU removal solutions [20], the investigated pipeline proves an interesting alternative method for image anonymization. Moreover, the proposed framework is computationally efficient because of its high parallelization potential. Indeed, inpainting is computed in parallel on different versions of the image, and results are merged at the end.

We also proposed a method to anonymize images by removing PRNU traces in a scenario in which the specific PRNU to be removed is assumed to be known. Despite state-of-the-art methods achieve better anonymization performance, we believe this work shows a different perspective on the topic, as the proposed solution makes use of a CNN in an uncommon fashion. Indeed, the CNN is seen as a parametric operator. CNN training is used to estimate CNN parameters by minimizing a loss function on a single image. From a different perspective, the proposed method works by overfitting a specific CNN to each input image. From the adversarial forensic point-of-view, results show an interesting aspect. If the denoising operators used for PRNU testing and within the

anonymization network match (i.e., DnCNN is used), images are strongly anonymized. If the analyst makes use of a different denoising operator for PRNU testing (i.e., the Wavelet-based one), anonymization may or may not be effective depending on the use correlation test. In reality, denoising operator matching is not needed by an attacker, given that the analyst is not informed about the possibility of an attack. If analysts know about possible attacks, they can use the symmetric test to avoid being completely fooled.

In the last section presented two PRNU-powered multimedia forensics applications. We proposed a method to combine different smartphone sensors to obtain a reliable, distinctive and easy-to-use fingerprint able to characterize each single device unequivocally. The challenge is to succeed in defining effective features and integrating them, though extracted from different sensors, to achieve a robust distinctiveness among diverse smartphones. The results obtained so far are encouraging, since mixing features from different sensors outperforms the classification obtained using the features from each sensor separately.

We finally presented a pipeline based on hash and camera fingerprint tailored to separate semantically similar videos while clustering together near duplicate ones. We validated the proposed algorithm on a set of more than 12 000 video sequences specifically designed in a challenging scenario, achieving promising results. From the computational point ov view, it is important to notice that a lot of data can be precomputed (e.g., hashes for rough detection, fingerprints for refinement, etc.) and only videos that pass the rough detection step are further processed, thus decreasing the computational burden.

Despite the robustness and precision of PRNU-based source identification systems, their deployment in real-world scenarios presents challenges that are evolving every day. To name a few, the effects of High Dynamic Range (HDR) interpolation algorithms and video coding on PRNU traces are still unknown, even though the number of devices acquiring, without an explicit user consensus, HDR pictures or short videos instead of steady images is constantly increasing.

4 Data-driven approaches in multimedia forensics

In this chapter we present two applications of data-driven approaches in the multimedia forensics field. In Section 4.1 we present a broad study on the use of Convolutional Neural Networks (CNN) for the problem of single vs double JPEG compression detection [23]. We explore the effect of several preprocessing and CNN architectures on different image patch sizes, and compare against state-of-the-art methods. In Section 4.2 we present a laser printer attribution system [24] based on the fusion of several CNN. Each CNN is trained to classify a specific letter with a specific preprocessing operation applied at the input. The effect of clean vs un-clean data input to the CNN is evaluated in terms of robustness. Both feature-based fusion and majority voting schemes are implemented to increase recognition accuracy.

4.1 Single vs. Double JPEG compression

In the last decades, due to the wide availability of easy-to-use imaging software, diffusion of tampered content has become a widespread phenomenon. Among the techniques developed by the image forensic community to fight this trend [1, 7], great attention has been devoted to methods analyzing JPEG traces [66, 165]. Indeed, every time an image is stored (e.g., at shooting time directly on the acquisition device, or after editing with processing tools), it is usually saved in JPEG format. Therefore, manipulated content often undergoes JPEG re-compression. Because of this fact, detection of double JPEG compression has received great attention in image forensics, and presence of tampering is often revealed by looking for the artifacts left by JPEG re-compression. However, depending on whether second JPEG compression grid is aligned or not with the one adopted by the first compression, different artifacts are introduced. For this reason, these two scenarios are often analyzed separately and are commonly referred to as aligned double JPEG (A-DJPEG) compression detection and non aligned double JPEG (NA-DJPEG) compression detection, respectively.

In many cases, manipulation takes place on limited parts of the image only. Therefore

DJPEG traces are only left on a limited number of pixels. For this reason, being able to detect DJPEG on small image patches proves paramount for localization of manipulated regions in image forgery detection problems. However, most of the techniques performing double JPEG detection in literature focus on estimating compression history of an image as a whole, whereas the localization of double compressed regions of relatively small size (i.e., possibly tampered regions) has been often overlooked and only addressed in some works. In this section we investigate the use of convolutional neural networks (CNNs) for the detection of A-DJPEG and NA-DJPEG even when working on small image patches (i.e., 64×64 pixel), which may be useful for forgery localization purpose.

4.1.1 Prior Work on Double JPEG Detection and Localization

It is well known that double JPEG compression leaves peculiar artifacts in the DCT domain, in particular, on histograms of block-DCT coefficients [166]. Accordingly, many proposed detection algorithms focus on the analysis of first order statistics of DCT coefficients. This is the case with the data-driven approach in [167], based on analysis of low-frequency block-DCT coefficients histograms, and many model-based approaches, e.g., the ones in [168–170] that rely on distribution of first (and sometimes second) significant digits (FSDs) in block-DCT coefficients and methods based on Benford-Fourier analysis [171, 172]. Data-driven detectors based on features derived from second-order statistics have also been proposed, e.g., [165]. A major drawback of many of these approaches is that they are designed to work on the whole image, i.e., to detect if an image has entirely undergone single or double JPEG compression and they fail to correctly classify small blocks or image patches, due to the difficulty of estimating the statistics in these cases. Therefore, they are not applicable in a tampering detection scenario, when only part of the image has been manipulated.

Among the algorithms performing localization, Lin et al. [76] exploit double quantization (DQ) effect on DCT coefficients' histograms to produce a likelihood map reporting tampering probabilities for each 8×8 block of the image. This method has been refined in [173] through use of an improved probability model. However, spatial resolution considered by the authors for good detection accuracy with these methods is 256×256, and performance drop significantly when smaller regions are considered. Besides, this method performs poorly when quality factor used for the first compression (i.e., QF1) is significantly larger than the second one (i.e., QF2). In [174], localization of spliced regions is achieved by using FSD features of block-DCT coefficients and employing a support vector machine (SVM) classifier. Recently, in [175], authors proposed a novel method that relies on a one-dimensional CNN, designed to automatically learn discriminant features

from DCT coefficients histograms. This approach outperforms both methods in [173] and [174], achieving good detection performance with small sized images up to 64×64 pixel. However, all the above approaches exploit the peculiar traces left by aligned DJPEG compression and then fail to detect double compression in the non-aligned case.

In the NA-DJPEG scenario, several other methods for detecting double compression have been proposed, relying on ad-hoc features extracted from both pixel domain [176, 177] and DCT domain [13, 178]. Specifically, in [177] authors proposed a method able to detect both aligned and non-aligned re-compression. The scheme works by combining periodic artifacts in spatial and frequency domains. Specifically, a set of features is computed to measure periodicity of blocking artifacts, which is altered when a NA-DJPEG compression occurs, and another set of features is used to measure periodicity of DCT coefficients, which is perturbed in presence of A-DJPEG. This approach for non-aligned re-compression detection is outperformed by [179]. Furthermore, in [180], Bianchi and Piva propose a forensic algorithm for tampering localization when DJPEG compression occurs, either aligned or not. The proposed scheme is as an extension of their analysis carried out in [173], where a unified statistical model characterizing JPEG artifacts in the DCT domain is considered. However, similarly to [173] (and [180]), this scheme works well as long as QF2 $>$ QF1; moreover, in order to achieve accurate detection, spatial resolutions lower than 256×256 pixel are not considered.

4.1.2 Contribution

Deep learning using convolutional neural networks (CNNs) [33, 34] has proved to be very powerful in many image classification problems, thus achieving considerable success in recent years also in steganalysis [46, 48, 63] and image forensics [10, 47, 72]. By using CNNs, the classical machine learning paradigm of manually extracting characteristic features from the data is replaced by the possibility of learning discriminant information directly from data.

Motivated by this recent trend, the goal of this work is to design CNN-based approaches able to classify single and double JPEG compressed images. Specifically, we are interested in working with small size images.

To the best of our knowledge, CNNs to perform double JPEG detection have been applied only in [175]. In this paper, a one-dimensional CNN is designed to take as input a feature vector built by concatenating DCT histograms. Since the network is fed with hand-crafted features (i.e., one-dimensional DCT histograms), the CNNs' capability of automatically learning from data is not addressed in that work.

We consider the case in which the image is directly given as input to the network, thus

fully exploiting self-learning capability of CNNs. Besides, our analysis is not limited to the case of aligned DJPEG compression, but we also consider the case of non-aligned double JPEG compression, in which the method proposed in [175] is not meant to work. Specifically, the contributions we present are detailed in the following. Concerning A-DJPEG detection:

- We refine the approach in [175] by showing that DCT histograms can be computed using common and readily available CNN layers, and that correlation among DCT histograms can be exploited to increase classification accuracy on small 64×64 images.
- We propose two alternative ways to perform detection based on CNNs with self-learned features directly from image pixels or noise residuals, showing the robustness of these algorithms in classifying images compressed with QFs different from those used for training.

Then, concerning NA-DJPEG compression:

- We compare the proposed CNN-based detectors against state-of-the-art solutions [169, 170, 179], showing that the CNN working on noise residuals significantly improves the performance especially on small 64×64 images.
- We confirm the robustness with respect to variations of QFs, showing that CNNs working on noise residuals are also able to correctly classify images compressed twice with the same QF.

Finally, when both A-DJPEG and NA-DJPEG are jointly considered we show that it is possible to use the same CNN-based methods to build a detector which works in the general case.

A strength of the proposed solutions, with respect to the most powerful state-of-the-art techniques (e.g., [175,179]), is that they are designed to work directly on the pixel values. Therefore, our algorithms can detect a double JPEG compression even when images are made available in bitmap or PNG format. This indeed can be seen as a simple yet effective antiforensic attack against the aforementioned methods which need access to the information in the JPEG bitstream (e.g., to read quantization tables or quantized coefficients).

4.1.3 Problem Formulation

JPEG is a lossy image transform coding technique based on block-wise Discrete Cosine Transform (DCT). In a nutshell, an image is split into 8×8 non-overlapping blocks,

each block is DCT transformed and quantized, then entropy coded and packed into the bitstream. Quantization is the operation causing information loss. Specifically, quantization is driven by pre-defined quantization tables scaled by a quality factor (QF). A lower QF indicates a stronger quantization, thus lower quality of the final decompressed image.

Double compression occurs when an image compressed with a quality factor QF1 is first decompressed and then compressed again with quality factor QF2. If no operations are applied between the two compression steps, 8×8 JPEG blocks of the first and second compressions are perfectly aligned, thus we speak of A-DJPEG compression. Conversely, when the second compression 8×8 grid is shifted with respect the previous one (e.g., due to cropping between first and second compression or to a cut and paste operation), we have a NA-DJPEG compression. Depending on the particular scenario, both A-DJPEG and NA-DJPEG may occur.

Our goal is to build a detector which is able to classify between single compressed and double compressed images. In other words, let H_0 correspond to the hypothesis of single compressed image, and H_1 to the hypothesis of image compressed twice. Given a $B \times B$ pixel image I, we want to detect whether H_0 or H_1 is verified, considering: i) only A-DJPEG case; ii) only NA-DJPEG; iii) both A-DJPEG and NA-DJPEG cases.

To solve this classification problem, we propose to use data-driven techniques based on CNNs. Specifically, starting from a standard supervised-learning pipeline, we propose three different architectures. The investigation of different approaches is motivated by the fact that aligned and non-aligned DJPEG compressions leave different footprints and then in principle cannot be detected in the same way.

4.1.4 Proposed Solutions

The proposed methodologies follow a common pipeline depicted in Figure 4.1 composed by two steps: training and test. During training, a database of labeled images is used to learn CNN parameters for the selected architecture. Accordingly, the CNN is fed with N pairs $\{\boldsymbol{I}_n, l_n\}$, $n \in [1, N]$, where $l_n = 0$ if image $\boldsymbol{I}_n$ verifies H_0 (single compressed), $l_n = 1$ if it verifies H_1 (double compressed). After training, the CNN outputs the learned model $\mathcal{M}$ containing all CNN parameters (e.g., filters, fully connected weights, etc.). Optionally, a pre-processing step (e.g., denoising) can be applied to the images, in order to turn images $\boldsymbol{I}_n$ into $\tilde{\boldsymbol{I}}_n$. When an image $\boldsymbol{I}$ is under analysis, it is fed to the trained CNN. The network outputs the probability of the image to verify whether H_0 is true or not. This probability (soft output) is converted to the estimated label $\hat{l}$ by thresholding (hard output). Clearly, if pre-processing is applied during training, it must be applied

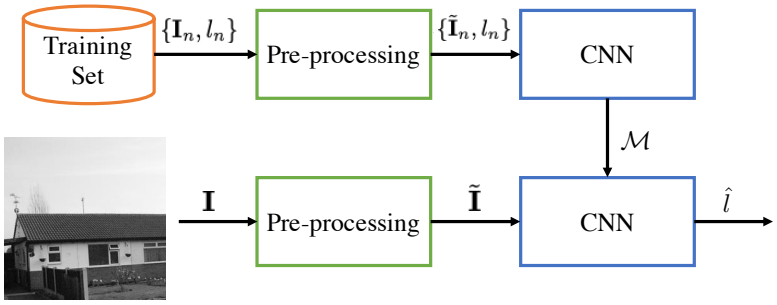


Figure 4.1: Pipeline common to the proposed solutions. CNN training (top) is performed using images $\boldsymbol{I}_n$ labeled with l_n. The CNN model $\mathcal{M}$ is then used for testing (bottom) a new image $\boldsymbol{I}$ and obtain the candidate label $\hat{l}$. Optional pre-processing might be applied to the images.

also during testing.

In the following we report the three investigated solutions, based on the above pipeline.

4.1.4.1 CNN in the Pixel Domain

The first investigated approach is based on the idea that properly designed CNNs should be able to automatically learn to distinguish between single and double compression by working directly on the image in the pixel domain. Encouraging results in this direction have been recently obtained in steganalysis field for classification of stego and cover images [62, 63].

In this case, $\boldsymbol{I}_n$ corresponds to the JPEG image in the pixel domain (decompressed) and $\tilde{\boldsymbol{I}}_n$ results from removing the average image $\frac{1}{N}\sum_{n=1}^{N}\boldsymbol{I}_n$ computed from the training set of images:

$$\tilde{\boldsymbol{I}}_n = \boldsymbol{I}_n - \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{I}_n. \tag{4.1}$$

The mean subtraction is customary done before CNN training to let the network work with almost-zero-average signals.

Regarding the CNN architecture, we resort to a slightly deeper variation of the well-known LeNet [33] developed for digits recognition, which has already been successfully exploited for forensic analysis [10, 45, 47]. This network architecture is depicted in Figure 4.2 (bottom part) and input-output size of each layer are reported in Table 4.1. $B \times B$ is the size of input grayscale image. Then, three convolutional layers (i.e., Conv-1, Conv-2 and Conv-3) apply stride 1 valid convolution with 30 filters 5×5 shaped. All of them are followed by a max-pooling layers (i.e., Pool-1, Pool-2 and Pool-3) with kernel 2×2. The first inner product layer (i.e., IP-1) reduces its input to 500 neurons and it is

Table 4.1: Reference CNN architecture parameters. Input-output relations for each layer are reported as function of the input image size $B \times B \times 1$.

Layer	Kernel size	Stride	Num. filters	Input Size	Output Size
Conv-1	5×5	1	30	$B \times B \times 1$	B-4 ×B-4 × 30
Pool-1	2×2	2	-	B-4 ×B-4 × 30	$B/2$-2 ×$B/2$-2 × 30
Conv-2	5×5	1	30	$B/2$-2 ×$B/2$-2 × 30	$B/2$-6 ×$B/2$-6 × 30
Pool-2	2×2	2	-	$B/2$-6 ×$B/2$-6 × 30	$B/4$-3 ×$B/4$-3 × 30
Conv-3	5×5	1	30	$B/4$-3 ×$B/4$-3 × 30	$B/4$-7 ×$B/4$-7 × 30
Pool-3	2×2	2	-	$B/4$-7 ×$B/4$-7 × 30	$B/8$-3 ×$B/8$-3 × 30
IP-1	-	-	500	$B/8$-3 ×$B/8$-3 × 30	500
ReLU-1	-	-	-	500	500
IP-2	-	-	2	500	2
SoftMax	-	-	-	2	2

followed by a ReLU non-linearity. Finally, the last fully connected layer (i.e., IP-2) reduces its input to 2 elements, i.e., one per class. SoftMax is used at the end to normalize IP-2 output to probability values.

Avoiding the use of deeper architectures, we are able to work even on small images. Indeed, after each convolutional and max-pooling layer, feature maps size is reduced by more than a half in the first and second dimensions. Therefore, starting from small input images it is not possible to go too deep, unless filter size and pooling strategies are changed. Moreover, the use of much deeper networks has proved to provide not many benefits in related forensic works [14].

4.1.4.2 CNN in Noise Domain

The second solution is based on the idea that additional pre-processing, aimed at removing irrelevant information (e.g., image content), may help the CNN in its training process. In order to expose double JPEG compression traces, we decided to rely on a denoising pre-processing operator. Then, the CNN input image $\tilde{\boldsymbol{I}}_n$ corresponds to the noise residual

$$\tilde{\boldsymbol{I}}_n = \boldsymbol{I}_n - \mathcal{F}(\boldsymbol{I}_n) \quad , \tag{4.2}$$

where $\mathcal{F}(\cdot)$ is the denoising operator described in [121], which relies on a spatially adaptive statistical model for the Discrete Wavelet Transform. The denoised image is predicted in the Wavelet domain by means of the minimum mean squared error (MMSE) estimation. This algorithm is widely used in forensics for its good capability of separating image content from noise [50, 105, 114]. With regard to the CNN architecture, we rely again on the one described in Table 4.1.

4.1.4.3 CNN Embedding DCT Histograms

The above solutions implicitly assume that DJPEG artifacts are exposed in the pixel domain. This is the case with non-aligned re-compressed images, which are characterized by a different behavior of blocking artifacts with respect to single JPEG compressed one [176, 177]. Conversely, when aligned re-compression is concerned, it is well known in the literature that peculiar traces are left in the DCT domain (specifically in the histogram DCT coefficient statistics), whereas traces left in the pixel domain are generally weaker. Therefore, our third proposed detection method relies on a CNN which automatically extract first order features from the DCT coefficients. We do not consider the case in which the image is block-wise transformed to the DCT domain and then directly fed to the CNN, because based on some preliminary experiments, we did not obtain good performances on small image patches ($B = 64$).

Despite this approach is similar to the one proposed in [175], we would like to stress that: i) we do not make use of DCT coefficients extracted from JPEG bitstream, rather we compute DCT with a CNN layer enabling us to work with decompressed images (i.e., our method still works if double JPEG images are stored in bitmap or PNG format); ii) we exploit a 2D-convolutional CNN, rather than a 1D one as done in [175], thus capturing possible correlation among DCT coefficient histograms; iii) our solution embeds histogram computation as part of the CNN, thus enabling fast and adaptive histogram computation using one of the many available GPU frameworks for CNN; iv) by embedding histogram computation in the CNN, we are able to also optimize the choice of quantization bins, rather than fixing it manually as in any hand-crafted approach.

Since this method does not make use of any pre-processing operation, $\tilde{\boldsymbol{I}}_n = \boldsymbol{I}_n$. Then, the used CNN can be thought as split into two parts as show in Figure 4.2: i) the former computes DCT coefficients histograms; ii) the latter, fed with these histogram, is the CNN described in Table 4.1, whose filters in convolutional layers are 3×3 rather than 5×5.

For the first part, the first step consists in obtaining the 2D DCT representation of each 8×8 image block. To this purpose, let us define $\boldsymbol{D}(c_1, c_2)$ as the $\frac{B}{8} \times \frac{B}{8}$ matrix containing the DCT coefficients at frequency (c_1, c_2) for each 8×8 image block. This can be easily computed with a convolutional layer as

$$\boldsymbol{D}(c_1, c_2) = \text{conv}_8(\boldsymbol{I}, \boldsymbol{H}(c_1, c_2)), \tag{4.3}$$

where $\text{conv}_8(\cdot, \cdot)$ computes the valid part of the 2D linear convolution using stride 8, and $\boldsymbol{H}(c_1, c_2)$ is the DCT base at (c_1, c_2) frequency. An example of $\boldsymbol{D}(c_1, c_2)$ is reported in

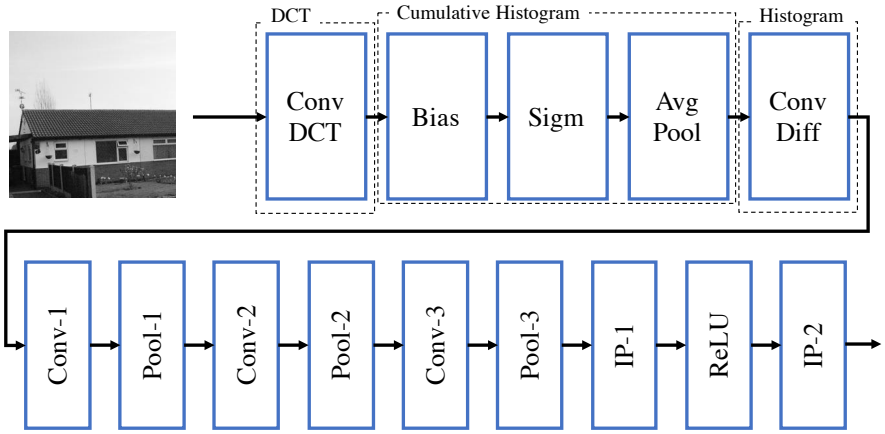


Figure 4.2: Pipeline of the CNN layers used by the third proposed method. On the top, the part devoted to DCT histogram computation. On the bottom, the CNN described in Table 4.1.

Figure 4.3.

At this point, for each frequency (c_1, c_2), we want to compute the histogram. To do so using common CNN layers, we first compute the cumulative histogram and then differentiate it. Specifically, to count the average number $c(c_1, c_2)_b$ of values in $\boldsymbol{H}(c_1, c_2)$ that are grater than a constant b, we resort to a series of bias, sigmoid and average-pooling layers obtaining

$$c(c_1, c_2)_c = \frac{B^2}{64} \sum_{i,j \in [0,7]} \text{sigmoid}\left[\gamma \cdot (D(c_1, c_2)_{i,j} - b)\right], \tag{4.4}$$

where the bias b is a constant value identifying a histogram bin boundary, γ is a gain (i.e., 10^6 in our experiments) used to expand the dynamic of $D(c_1, c_2)_{i,j} - b$ (i.e., to obtain very high values for $D(c_1, c_2)_{i,j} > b$ and very low values for $D(c_1, c_2)_{i,j} < b$), sigmoid$(\cdot)$ turns very high and very low input values into 0 or 1, and the average-pooling layer performs the sum and normalization for $\frac{B^2}{64}$. In other words, $c(c_1, c_2)_b$ is the b-th cumulative histogram bin for DCT coefficient (c_1, c_2). Examples of these signals are depicted in Figure 4.3.

The histogram for each (c_1, c_2) coefficient can be obtained using a convolutional layer that computes

$$\boldsymbol{z}(c_1, c_2) = \text{conv}_1(\boldsymbol{c}(c_1, c_2), [1, -1]), \tag{4.5}$$

where conv computes 1D convolution, and the filter $[1, -1]$ acts as differentiator in the b-th direction. Differently from [175], we do not assume to already have access to quantized DCT coefficients. Therefore, the set of b values use to construct histograms is not known

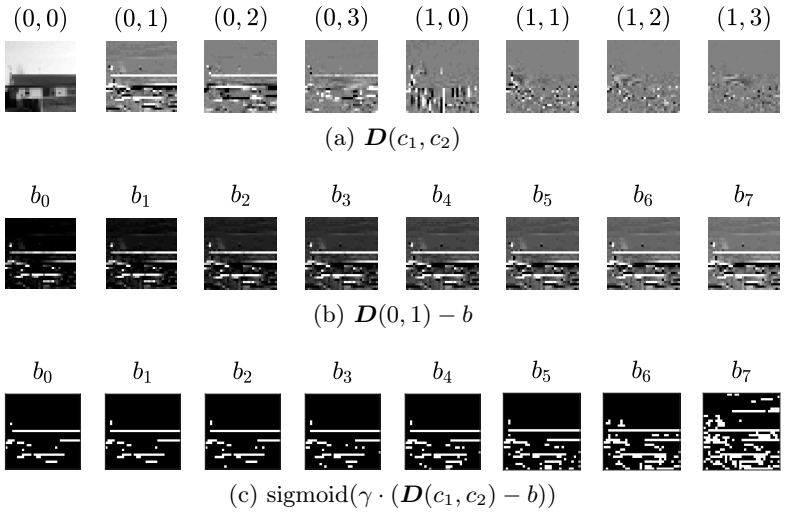


Figure 4.3: Outputs of CNN layers devoted to histogram computation: (a) output of the DCT layer $\boldsymbol{D}(c_1, c_2)$ for nine different pairs (c_1, c_2); (b) output of the bias layer $\boldsymbol{D}(0, 1) - b$ for $(c_1, c_2) = (0, 1)$ and different b values; (c) output of sigmoid layer sigmoid$(\gamma \cdot (\boldsymbol{D}(c_1, c_2) - b))$ for $(c_1, c_2) = (0, 1)$ and different b values.

and must be sought. An example of obtained histogram and its cumulative version is reported in Figure 4.4.

Once all histograms $\boldsymbol{z}(c_1, c_2)$ for all considered DCT frequency pairs (c_1, c_2) have been computed in parallel by the CNN, they are concatenated into a 2D matrix $\boldsymbol{Z}$, where each row represents a histogram bin b, and each column represents a frequency pair (c_1, c_2). This matrix (i.e., the output of ConvDiff layer of Figure 4.2) can be considered as an image, fed as input to the CNN pipeline defined in Table 4.1.

4.1.5 Dataset Construction

In order to thoroughly validate the proposed solutions, we generated a set of training and test datasets of single and double compressed images at different resolutions and with different quality factors, for a total amount of more than 3M images. All datasets are built starting from images of RAISE database [181]. This is a collection of more than 8 000 uncompressed real-world images of high resolution taken from different cameras. Images have been first converted to grayscale, then randomly cropped in order to obtain smaller resolution images used in our tests. Attention is paid to split into only one set (training or validation) all cropped portions coming from the same original image. All sets are balanced, i.e., they contain the same number of single and double JPEG images.

Training sets have been created in the following cases: i) $B = 64, 256$; ii) aligned and non-aligned DJPEG. Each set contains between 280k and 300k image patches. For each scenario, the image set is built as it follows: for the first class (H_0), images of size

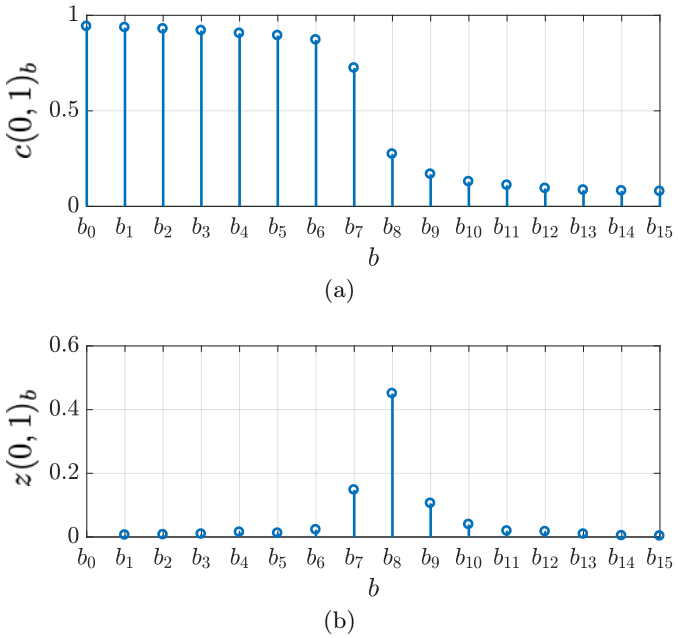


Figure 4.4: Example of (a) cumulative histogram $c(c_1, c_2)_b$ and (b) histogram $z(c_1, c_2)_b$, for $(c_1, c_2) = (0, 1)$ and $b \in [b_0, b_{15}]$. These are the output of average pooling and derivative convolutional layer fixing $(c_1, c_2) = (0, 1)$, respectively.

$B \times B$ are single compressed with quality factor QF; for the second class (H_1), double compressed images are built by coding $B \times B$ images first with various QF1 and then with QF2. For a meaningful analysis, we take QF = QF2 as done in [175].

To build double compressed images for the non-aligned case, we start from images of size $B' \times B'$ with $B' \geq B + 7$. Then, after the first compression with $QF1$, images are shifted by a random quantity (r, c), $0 < r, c < 7$, and cropped to the size $B \times B$, before being compressed again with $QF2$, thus simulating grid misalignement. In all our experiments, we consider three possible values for QF2, that is 75, 85 and 95, whereas QF1 $\in \{50, 60, 70, 80, 90\}$ for the first two QF2 values and QF1 $\in \{60, 70, 80, 90, 98\}$ for the last one. Table 4.2 reports the breakdown of all these training datasets. We denote with $\bar{\mathcal{D}}$ datasets for the aligned DJPEG case and with $\hat{\mathcal{D}}$ datasets for non-aligned JPEG scenario. Superscripts indicate the adopted QF2 (i.e., 75, 85 or 95), whereas subscripts indicate image size (i.e., $B = 64$ or 256).

Validation datasets have been created to evaluate: i) detection accuracy under normal working conditions, i.e., the ability of classifying test images built under the same conditions of training, and also; ii) generalization capability, that is, the ability of classifying images even when they are not perfectly compliant with the used training set. To this purpose, we generated different sets of double JPEG images with many different

Table 4.2: Datasets used for training. All datasets are balanced in both classes and QF pairs.

Datasets	I **Size**	**QF**1	**QF**2	**Alignment**	# **Train**	# **Val**
$\bar{\mathcal{D}}_{256}^{(75)}/\bar{\mathcal{D}}_{256}^{(85)}$	256x256	50,60,70,80,90	75/85	A	280k	30k
$\hat{\mathcal{D}}_{256}^{(75)}/\hat{\mathcal{D}}_{256}^{(85)}$				NA		
$\bar{\mathcal{D}}_{256}^{(95)}$		60,70,80,90,98	95	A		
$\hat{\mathcal{D}}_{256}^{(95)}$				NA		
$\bar{\mathcal{D}}_{64}^{(75)}/\bar{\mathcal{D}}_{64}^{(85)}$	64x64	50,60,70,80,90	75/85	A	300k	30k
$\hat{\mathcal{D}}_{64}^{(75)}/\hat{\mathcal{D}}_{64}^{(85)}$				NA		
$\bar{\mathcal{D}}_{64}^{(95)}$		60,70,80,90,98	95	A		
$\hat{\mathcal{D}}_{64}^{(95)}$				NA		

(QF1, QF2) pairs and single JPEG images with the corresponding QF2. Specifically, in addition to the same pairs used for training, we consider some new pairs where QF1 or QF2 deviates from the values used for training. Each set contains 3 000 single compressed images and 3 000 double compressed ones. As for training, validation sets are built for the case $B = 64$ and 256, with either aligned or non-aligned DJPEG.

As commonly done to evaluate the performance with data-driven approaches, detection accuracy is measured over the same (QF1, QF2) pairs used for training. Then, to test their generalization capability, we also measure the performance of the detectors with respect to (QF1, QF2) pairs never used for training.

4.1.6 Evaluation Methodology

In order to fairly evaluate all CNN-based considered approaches, we devised a common training-validation strategy. All CNNs have been trained using stochastic gradient descent (SGD) algorithm with batch size (i.e., number of images used for each SGD iteration) set to 128. Momentum was set to 0.9. Learning rate was set to 0.01 for 64×64 images and 0.001 for 256×256 images, and was progressively decreased with exponential decay at each iteration. The maximum amount of epochs (i.e., number of times the CNN sees all training data) was set to 30 to ensure network convergence. Initialization of CNN parameters have been performed using the method devised in [182]. As best CNN trained model, we always selected the one at the epoch with minimum validation loss in order to avoid overfitting. All experiments have been run exploiting Caffe framework [183] on a workstation equipped with an Intel(R) Xeon(R) CPU E5-2630 v2 @ 2.60GHz with 64GB of RAM and one NVIDIA Titan-X GPU.

The results are provided in terms of accuracy, namely the percentage of correctly classified single and double JPEG images in the validation dataset. We use notation $\mathcal{C}_{\text{pix}}$ to refer to the CNN-based detector in the pixel domain, $\mathcal{C}_{\text{noise}}$ for the one in the noise domain, and $\mathcal{C}_{\text{hist}}$ for the case of CNN embedding DCT histogram computation. Con-

cerning parameters of the latter, we made use of all the AC DCT frequencies. Histograms have been computed using 101 integer bins initialized with $b \in [-50, 50]$.

4.1.7 Aligned Double JPEG

It is well known that the performance of supervised machine learning techniques strongly depends on the amount of data used for training. In order to assess the dependency between number of images used for training and detection accuracy in our case, Figure 4.5(a) shows the results achieved with $\mathcal{C}_{\text{pix}}$ in the most difficult scenario with small patches ($B = 64$) and strong second quantization (QF = 75). To get the plot, the network is trained on different percentages of training images from $\bar{\mathcal{D}}_{64}^{(75)}$. We see that, when 10% of the dataset is used for training, accuracy is below 0.75. However, when more than 70% of training data is used, accuracy saturates around 0.82. Therefore, using the whole training dataset, we are sure that we are not experiencing losses due to insufficient amount of training data.[1]

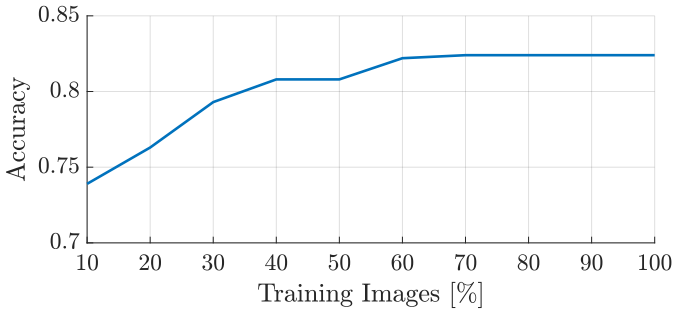
In order to assess the effect of CNN architecture deepness, we trained five CNNs with increasing number of Conv-Pool layer pairs on a subset of the whole dataset. Results reported in Figure 4.5(b) show how the selected architecture almost saturates the achievable performance in terms of accuracy.

To assess the performance of the proposed approaches for aligned double JPEG detection, we compare them to the state-of-the-art techniques in [175], [169] and [170], denoted respectively as WZ, KH and TR in plot legends. We select [175] as one of the baseline for two reasons: i) it is shown to outperform previously existing state of the art detectors, e.g., [167, 168, 173, 174]; ii) to the best of our knowledge, it is the only method based on CNNs, thus being a natural comparison for our methods.

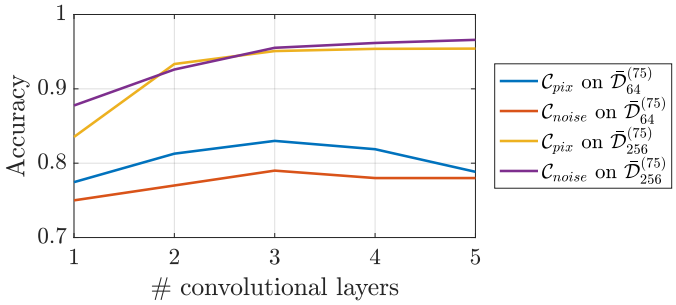
Figure 4.6 reports results obtained training all proposed CNNs in the various cases, i.e., on the datasets $\bar{\mathcal{D}}_{256}^{(75)}$, $\bar{\mathcal{D}}_{256}^{(85)}$, $\bar{\mathcal{D}}_{256}^{(95)}$, $\bar{\mathcal{D}}_{64}^{(75)}$, $\bar{\mathcal{D}}_{64}^{(85)}$ and $\bar{\mathcal{D}}_{64}^{(95)}$. Results for $B = 256$ show that the proposed $\mathcal{C}_{\text{hist}}$ architecture achieves equal or better performance with respect to all baseline methods. This is due to the fact that hand-crafted features exploited in [175] are very distinctive, especially when large images are concerned.

With small patches ($B = 64$) all algorithms suffer when QF2 $\cong$ QF1 (this case is addressed in the literature by specific methods tailored for the purpose, e.g., [184]) and QF2 $<$ QF1, as a stronger second compression tends to mask artifacts left by the first one. However, on 64×64 patches, $\mathcal{C}_{\text{hist}}$ is the one with the best performance and always outperforms state-of-the-art methods on average.

[1] It is worth pointing that the other proposed solutions, i.e., $\mathcal{C}_{\text{noise}}$ and $\mathcal{C}_{\text{hist}}$, usually need less training images to converge.

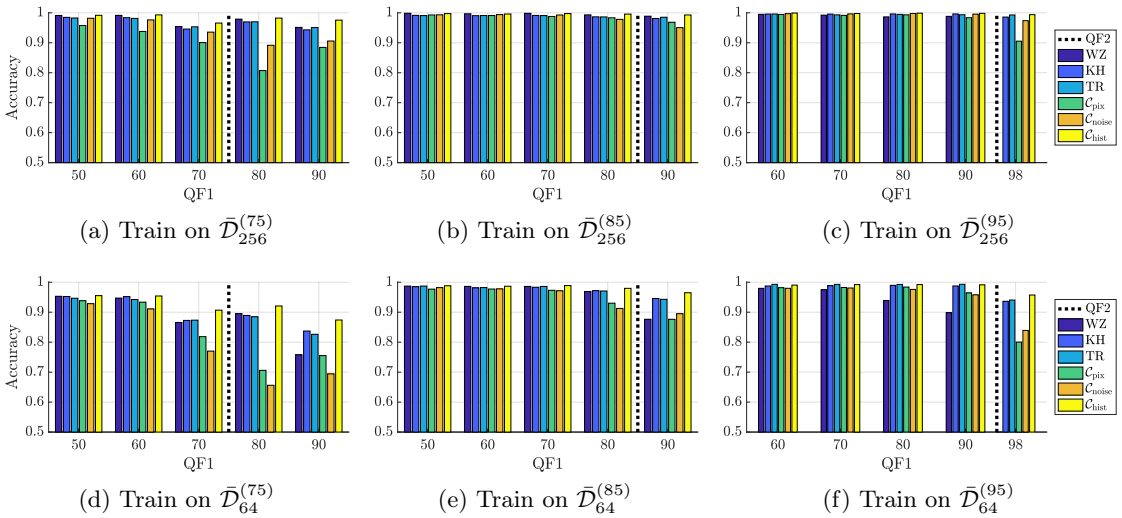


(a) Impact of training set size on A-DJPEG detection accuracy using $\mathcal{C}_{\text{pix}}$.



(b) Impact of CNN depth on A-DJPEG detection accuracy using $\mathcal{C}_{\text{pix}}$ and $\mathcal{C}_{\text{noise}}$.

Figure 4.5: Impact of training set size and number of CNN layers.



(a) Train on $\bar{\mathcal{D}}_{256}^{(75)}$ (b) Train on $\bar{\mathcal{D}}_{256}^{(85)}$ (c) Train on $\bar{\mathcal{D}}_{256}^{(95)}$

(d) Train on $\bar{\mathcal{D}}_{64}^{(75)}$ (e) Train on $\bar{\mathcal{D}}_{64}^{(85)}$ (f) Train on $\bar{\mathcal{D}}_{64}^{(95)}$

Figure 4.6: Aligned DJPEG compression detection accuracy against baselines WZ [175], KH [169], and TR [170]. Dashed black line indicates the considered QF2.

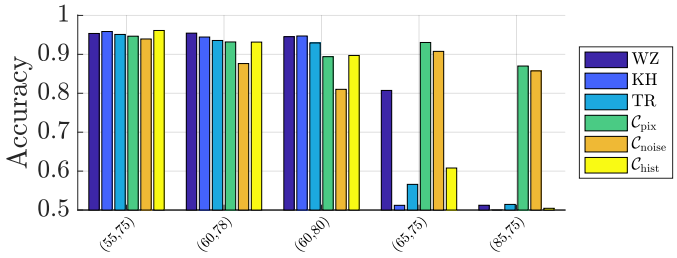


Figure 4.7: Sensitivity analysis for aligned DJPEG compression detection when QF2 = 75. Image size is 64×64.

Concerning the proposed methods, $\mathcal{C}_{\text{hist}}$ always outperforms $\mathcal{C}_{\text{pix}}$ and $\mathcal{C}_{\text{noise}}$. This is also expected, as aligned DJPEG traces are better exposed in the DCT domain, rather than the pixel domain. Nonetheless, a part when QF1 and QF2 are very close, also $\mathcal{C}_{\text{pix}}$ and $\mathcal{C}_{\text{noise}}$ allow to achieve accuracy greater than 0.70 on small images.

Regarding generalization capability, Figure 4.7 shows the accuracy achieved by all CNNs trained on the most difficult scenario with QF = 75 and small images ($B = 64$).

The methods based on DCT histograms or Benford law (i.e., $\mathcal{C}_{\text{hist}}$ and baselines WZ, KH, TR) suffer to recognize aligned DJPEG for values of QF1 different from those used during training when they are close to QF2, and completely fail when these QF1s are larger than QF2.

Contrarily, the methods relying on pixel analysis (i.e., $\mathcal{C}_{\text{pix}}$ and $\mathcal{C}_{\text{noise}}$) show greater robustness to changes in $(\text{QF1}, \text{QF2})$.

To further explore this fact, Table 4.3(a) shows the behavior of $\mathcal{C}_{\text{noise}}$ trained on $\bar{\mathcal{D}}_{64}^{(75)}$ and $\bar{\mathcal{D}}_{256}^{(75)}$ and tested on images with several different $(\text{QF1}, \text{QF2})$ pairs (similar results hold for $\mathcal{C}_{\text{pix}}$). Similarly, Table 4.3(b) reports the accuracy results with $\mathcal{C}_{\text{noise}}$ trained on $\bar{\mathcal{D}}_{64}^{(85)}$ and $\bar{\mathcal{D}}_{256}^{(85)}$. We notice that, by varying QF1, results are perfectly in line with those achieved with matched QF pairs. Good results are also obtained with different QF2s, a part for the case of much higher QF2.

This behavior is not surprising, since compression with high QF2 leaves few traces on images compressed at lower quality, hence detecting a DJPEG compression in these cases is hard when such examples are not included in the training set.

To conclude the analysis of this section, although on one side CNNs based on a strong hand-crafted modeling assumption (as baseline [175] and $\mathcal{C}_{\text{hist}}$) allow to achieve the best accuracies, the ones based on the analysis of the pixel image (i.e., $\mathcal{C}_{\text{pix}}$ and $\mathcal{C}_{\text{noise}}$) prove to be more robust to perturbations of QF1 and QF2 with respect to the values used for training, which is paramount every time the algorithm works in the wild.

Table 4.3: Sensitivity of $\mathcal{C}_{\text{noise}}$ to variations of QF1 and QF2 for aligned DJPEG detection. For any pair, only one between QF1 and QF2 is common to images used in the training set (reported in **bold**).

Testing (QF1, QF2)	$B = 64$	$B = 256$
(55, **75**)	0.925	0.982
(65, **75**)	0.880	0.981
(85, **75**)	0.820	0.952
(**60**, 78)	0.900	0.917
(**70**, 78)	0.810	0.907
(**60**, 80)	0.860	0.810
(**70**, 80)	0.790	0.800

(a) Train on $\bar{\mathcal{D}}_B^{(75)}$, $B \in \{64, 256\}$.

Testing (QF1, QF2)	$B = 64$	$B = 256$
(55, **85**)	0.963	0.994
(65, **85**)	0.960	0.993
(75, **85**)	0.923	0.978
(**70**, 88)	0.860	0.914
(**80**, 88)	0.640	0.656
(**70**, 90)	0.718	0.687
(**80**, 90)	0.500	0.510

(b) Train on $\bar{\mathcal{D}}_B^{(85)}$, $B \in \{64, 256\}$.

4.1.8 Non-aligned Double JPEG

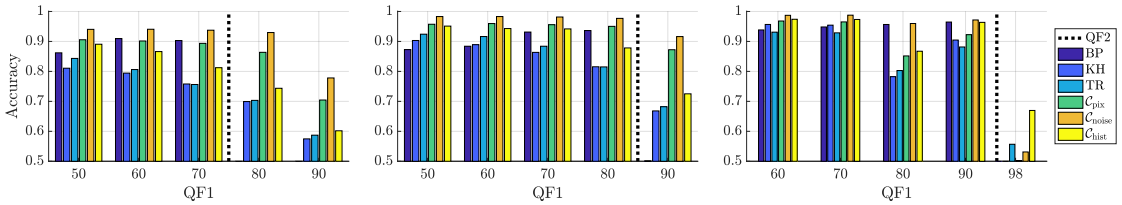
When DJPEG compression occurs with misalignment between the grids, detectors in the previous section trained on aligned data do not work anymore, getting an accuracy which is around 0.5. To evaluate the performance of our method for NA-DJPEG detection, we re-train the detectors in the misaligned case. In this case, not surprisingly, the algorithm in [175] (WZ) does not work. Indeed, the features extracted by this method, i.e., the DCT histograms, are particularly distinctive only when the second compression is aligned with the first one (the typical peak and gap artifacts shows up in the DCT histograms). Therefore, we select the well-known algorithm for NA-DJPEG detection proposed in [179], denoted as BP, as additional baseline in this case.

Figure 4.8 shows the performance of all proposed techniques and baselines for QF2 = 75, 85 and 95 with image size 64×64 and 256×256. It is known that BP does not work when QF1 > QF2. Besides, the accuracy significantly drops for small images, especially in the case QF1 $\simeq$ QF2. Concerning our methods, not surprisingly, our solution $\mathcal{C}_{\text{hist}}$ shows poor performance with respect to $\mathcal{C}_{\text{pix}}$ and $\mathcal{C}_{\text{noise}}$. Indeed, similarly to [175], the traces in the DCT domain that $\mathcal{C}_{\text{hist}}$ looks at are weak in the non-aligned case.

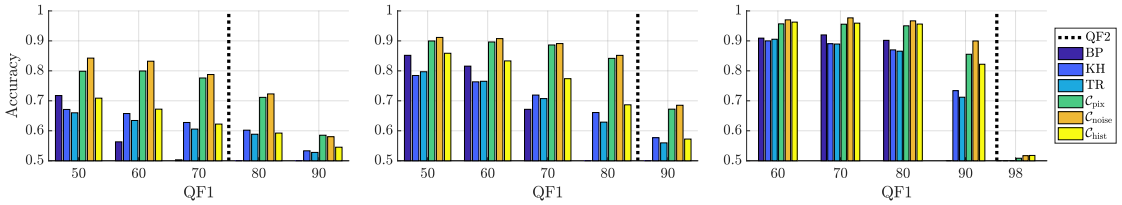
On the other hand, CNNs designed to work in the pixel domain show good detection performance even for small images (i.e., 64×64). From these results, we see that the detector based on $\mathcal{C}_{\text{noise}}$ always outperforms state of the art.

Concerning network sensitivity to QF pairs different from those in the training set, Table 4.4 shows the results obtained with our best method $\mathcal{C}_{\text{noise}}$ for both QF1 = 75 and 85, and image sizes. As for the aligned scenario, $\mathcal{C}_{\text{noise}}$ enables good detection accuracy, the only critical cases being those with much higher QF2. It is interesting to notice that $\mathcal{C}_{\text{noise}}$ is able to detect non-aligned DJPEG compression with good accuracy also in the very challenging scenario in which QF1 = QF2.

When double compression occurs with QF2 = 95 and QF1 > 95, the detector fails and



(a) Train on $\hat{\mathcal{D}}_{256}^{(75)}$ (b) Train on $\hat{\mathcal{D}}_{256}^{(85)}$ (c) Train on $\hat{\mathcal{D}}_{256}^{(95)}$



(d) Train on $\hat{\mathcal{D}}_{64}^{(75)}$ (e) Train on $\hat{\mathcal{D}}_{64}^{(85)}$ (f) Train on $\hat{\mathcal{D}}_{64}^{(95)}$

Figure 4.8: Non-aligned DJPEG compression detection accuracy against baselines BP [179], KH [169], and TR [170]. Dashed black line indicates the considered QF2.

Table 4.4: Sensitivity of $\mathcal{C}_{\text{noise}}$ to variations of QF1 and QF2 for non-aligned DJPEG detection. Test and training images have only QF1 or QF2 in common (reported in **bold**).

Testing (QF1, QF2)	$B = 64$	$B = 256$
(55, **75**)	0.816	0.876
(65, **75**)	0.805	0.866
(75, **75**)	0.764	0.842
(85, **75**)	0.674	0.776
(**60**, 78)	0.777	0.845
(**70**, 78)	0.765	0.830
(**60**, 80)	0.723	0.794
(**70**, 80)	0.720	0.790

(a) Train on $\hat{\mathcal{D}}_{B}^{(75)}$, $B \in \{64, 256\}$.

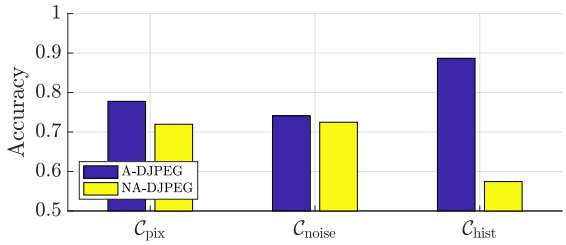
Testing (QF1, QF2)	$B = 64$	$B = 256$
(55, **85**)	0.897	0.972
(65, **85**)	0.878	0.972
(75, **85**)	0.865	0.961
(85, **85**)	0.793	0.954
(**70**, 88)	0.751	0.786
(**80**, 88)	0.738	0.785
(**70**, 90)	0.650	0.610
(**80**, 90)	0.634	0.600

(b) Train on $\hat{\mathcal{D}}_{B}^{(85)}$, $B \in \{64, 256\}$.

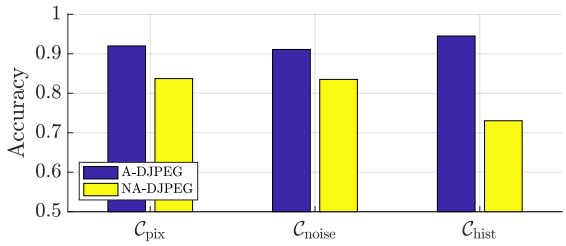
the images are misclassified half of the time. Experiments show that even if we train our methods to detect this specific case, the accuracy does not go above 66%, thus confirming that the misalignment between the 8×8 compression grid tends to remove completely the traces, which in this case were already very weak in the aligned case, and then makes the detection very challenging.

4.1.9 Aligned and Misaligned Double JPEG

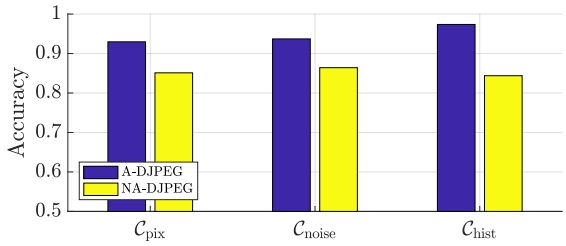
Since it is usually not known a-priori whether double compression is aligned or not, it is relevant to be able to detect both A-DJPEG and NA-DJPEG. To this purpose, we trained the proposed architectures on a dataset obtained by the union of the one



(a) Train on $(\bar{\mathcal{D}}_{64}^{(75)} \cup \hat{\mathcal{D}}_{64}^{(75)})$



(b) Train on $(\bar{\mathcal{D}}_{64}^{(85)} \cup \hat{\mathcal{D}}_{64}^{(85)})$



(c) Train on $(\bar{\mathcal{D}}_{64}^{(95)} \cup \hat{\mathcal{D}}_{64}^{(95)})$

Figure 4.9: DJPEG compression detection accuracy tested separately on aligned and misaligned cases, when training is performed on a mixed dataset. Image size is 64 and QF2 $= 75, 85$.

used for A-DJPEG, namely $\bar{\mathcal{D}}$, and the one used for NA-DJPEG, namely $\hat{\mathcal{D}}$. For the experiments of these section, we considered the most challenging scenario with small images ($B = 64$). Figure 4.9 shows the performance of the CNN-based detectors in terms of average accuracy computed separately on A-DJPEG and NA-DJPEG images. The average is taken over all the QF pairs used for training. As expected from the previous analysis, $\mathcal{C}_{\text{hist}}$ tends to learn better characteristics of aligned DJPEG and performs poorly in non-aligned case. Conversely, $\mathcal{C}_{\text{pix}}$ and $\mathcal{C}_{\text{noise}}$ are more stable solutions being able to detect with almost the same accuracy both A-DJPEG and NA-DJPEG images.

Driven by the accurate performance of $\mathcal{C}_{\text{hist}}$ on A-DJPEG compression, we also investigated an alternative solution according to which the detection for the mixed case is obtained by fusing the outputs of our best CNN-based detectors for the aligned and non-aligned case, through the use of a binary classifier. Specifically, we considered the

(a) (b)

Figure 4.10: A-DJPEG (a) and NA-DJPEG (b) localization example of a compressed central region with $QF2 = 95$ and $QF1 = 80$. $\mathcal{C}_{\text{hist}}$ is used for the aligned case while $\mathcal{C}_{\text{noise}}$ for the non aligned case. Actual forged region lies inside the yellow rectangle.

output provided by $\mathcal{C}_{\text{hist}}$ trained on A-DJPEG images, and the output of $\mathcal{C}_{\text{noise}}$ trained in the NA-DJPEG case, as feature vector. By feeding this feature vector to a binary classifier (i.e., a random forest in our case), it is possible to further increase the final accuracy in the mixed case by up to 2%. However, other solutions and fusing strategies might be investigated. We leave a thorough investigation of this case to future studies.

4.1.10 Localization

Given the good performance achieved on small patches, our method can be applied on sliding windows to localize possible tampering regions in images. This can be done, e.g., by dividing the image into overlapping blocks of size 64×64 with stride 16×16. Each block is fed to the CNN (after a pre-processing step for the case of $\mathcal{C}_{\text{noise}}$) and the softmax output is used as an estimation of the probability that the block is double compressed. Figure 4.10 shows the results of double compression localization of a central region, bounded in yellow, in A-DJPEG and NA-DJPEG scenarios with $QF2 = 95$ and $QF1 = 80$, when $\mathcal{C}_{\text{hist}}$ is used for the former case and $\mathcal{C}_{\text{noise}}$ for the latter case. Both examples show that red-shaded blocks, i.e. those for which the probability of being double compressed is higher, are mainly inside the expected central region.

4.1.11 Conclusions

In this section we explored the use of CNNs for double JPEG compression detection problem in the case of aligned and non-aligned recompression. Specifically, three different solutions are investigated: in one of them, the CNN is based on hand-crafted features extracted from the images; in the other two, the CNN is trained directly with the images

and the denoised versions, then features are self-learned by the CNN itself.

Results show that CNN based on hand-crafted features allow to achieve better accuracies in the case of A-DJPEG. For the NA-DJPEG instead, the CNN based on self-learned features applied to the image noise residuals is shown to outperform the state of the art in every tested scenario. Good performance are achieved even in the difficult cases in which the second quality factor is larger than the first and over small images, thus paving the way to the application of the techniques to tampering localization. Besides, CNN based on self-learned features prove very robust to deviations between training and test conditions. Additionally, some preliminary experiments show the proposed CNN-based methods can also be successfully applied to simultaneously detect an aligned or non-aligned DJPEG compression.

We designed our methods by assuming that no processing operation occurred in the middle of the two compression stages. Although this is a common assumption to D-JPEG detection approaches in the literature, in real applications, some intermediate processing might be applied. In view of this, we made some preliminary tests to check if and at which extent the D-JPEG detector is robust to basic processing operations[2]. The tests show that good resilience is achieved on the average with respect to histogram enhancement operations (accuracy around 85%) and cropping (80%), which just introduces a 8×8 grid desynchronization as a main effect. On the other side, the performance with respect to filtering operation are poor (62% of accuracy in the case of a light blurring, performed with a 3×3 Gaussian smoothing kernel with variance $\sigma^2 = 1$). The classification fails in the case of geometric transformation, e.g., resizing (around 30%).

4.2 Laser Printer Attribution

Printed documents are found everywhere. From simple documents available today such as homeworks and warnings, to more crucial ones such as contractual clauses and scientific articles, a printer is always involved, being it a dot matrix, dye-sublimation, thermal, ink-jet or laser. The last one has been the choice of domestic users and offices in the last decade because of its speed, quality of printing and decreasing price.

However, with this massive access to printing devices, a new threat has also emerged: the use of laser printers for criminal intentions. Additional contractual clauses inexistent before, child pornography and animal abuse photos, life threatening letters, illegal correspondence, terrorist plots, fake currency and fake documents can now be easily printed

[2]Results are referred to $\mathcal{C}_{\text{noise}}$ trained on both aligned and misaligned D-JPEG compressed images (Section 4.1.9) with QF2 = 85 and $B = 256$.

by anyone. Hence, providing ways of pinpointing printing ownership of documents is paramount, mainly to link them to criminals. Also, linking a document to a printer is another way of authenticating official documents.

Several approaches have been proposed for this task in the literature. Some techniques are based on laboratory analysis of the actual used paper [185, 186]. However, these methods can damage or even destroy investigated documents as chemical and physical procedures are involved. Another branch of approaches exploits the so called extrinsic signatures, which are characteristic footprints printed on documents, either visible or not to the naked eye. These signatures can be embedded into printed material by modifying the printing process to encode some sort of source identification [187]. This can be done, for example, using watermarks, pulse width modulation, QR-codes or security deterrents [188–191]. Recently, it has been reported that some printers encode, on the printed pages, some provenance information using tiny yellow dots spread over the printing material, no matter if the document is colored or not [192]. The limitation of these approaches is the fact that they do not represent a gold standard followed by the whole industry, and an expert user can change the printer's firmware maliciously.

Finally, another group of methods aims at solving printer attribution in a non-invasive (i.e., preserving the original document) blind fashion. This means these methods do not rely on printer information embedded into documents. Rather, they rely on signatures left by mechanical imperfections specific of printers that can be searched for on the printed material [193–196]. These techniques use computer vision and machine learning approaches applied to scanned versions of suspected documents. More specifically, existing methods for text (non-colored) documents make use of hand-crafted features generated by an initial assumption about printing imperfections. These features are then extracted from a limited part of the data (e.g., one symbol or letter of the raw text) [194, 197–199] and fed to supervised classifiers for reaching a decision upon the printer source of the document.

As the use of engineered features has been recently challenged by feature learning paradigms in many detection and recognition tasks [37], in this work, we present a data-driven printer attribution approach. This is the first deep learning solution for laser printer attribution that uses several Convolutional Neural Networks (CNNs) in parallel, extracting meaningful discriminative patterns straight from the analyzed documents instead of using ordinary feature engineering. Our approach exploits the advantages of back-propagation procedures, commonly used in CNNs, to automatically learn discriminant features from a set of existing training documents. It also uses different data representations to better identify printing pattern artifacts on an input printed character,

further enhancing the characterization process and the analysis of provenance of a printed document (attribution task). Finally, we apply a late-fusion paradigm to integrate the classification outcomes coming from different letters within the same document.

The proposed approach is tailored to blind laser printer attribution for grayscale text documents. This means we do not rely upon any prior information such as inserted watermarks, and the traces we exploit can be extracted by the analysis on sets of letters. As the proposed method builds upon a machine learning framework, we assume the availability of a set of training documents as for any other supervised learning approach in the literature [195, 200]. More specifically, we consider that the only available data are scanned versions of: (i) the questioned document, and; (ii) a set of training documents coming from a set of suspect or candidate printers. The available training documents are considered to be printed with the same font and approximately the same font-size of the document under analysis. Moreover, we assume that some training documents actually come from the printer used to generate the document under investigation. In this setup, we consider that all the documents have been scanned with the same scanner, in order to avoid introducing any additional bias.

Notice that, even though these hypotheses may seem strict, we are not bounding our method neither to work with a single font and font-size, nor to work with a fixed character. Moreover, in courts of law, it is common that: (i) the analyst has direct access to many documents printed with the suspect printer, or; (ii) the analyst has access to the suspect printer itself. In the first case, as the commonly used fonts and sizes for official documents are not many, the analyst has a high probability of owning sufficient data with the same font and (approximate) size. In the second case, the analysis is even simpler, as the analyst can print as many documents he/she wants, with any font and size.

In summary, the main contributions of this section are:

1. The design and development of an ad-hoc CNN for laser printer attribution based on the analysis of small patches representing text characters. The network is characterized by a small amount of parameters, thus allowing a fast yet reliable training with a limited set of labeled data.

2. The use of CNNs on multiple representations of the same character to learn complementary features that are fused together for an increased recognition accuracy.

3. The use of a late-fusion paradigm to merge results coming from the analysis of different characters within the same document. In this way, each character is

classified separately, and individual results contribute to the final document label. This is useful especially for documents containing repetitions of some letters.

4.2.1 Literature solutions for laser printer attribution

Laser Printers (LPs), differently from ink-jet printers, use a dry painting process based on the electromagnetic attraction of sooty powders inside a toner and the paper to be printed, in a process conducted by modifying charges on a light-sensitive revolving drum by a laser light source reflected by mirrors. The laser printer process occurs, in a nutshell, by charging this drum by a laser reflected by a mirror, which attracts the positive charged toner. Finally, the paper attracts the toner and a fusing process, by heat, joins the toner to the paper.

The intrinsic characteristics that can be seen on printed pages during this process are generated by imperfections in the manufactured parts of LPs, such as the leak of electric charges in some parts of the drum, different patterns of mirrors angle for different manufacturers, different speed of the revolving drum, among others. One of these intrinsic characteristics is called banding and is the most considered by the literature. Banding is defined as light and dark lines in a perpendicular direction to where the paper is moved inside the printer [188,201]. Different brands are characterized by almost unique banding frequencies on different models of printers [195]. Several techniques in the literature have been focused on detecting such banding artifacts. Most of them can be divided in approaches focused on color documents (images) and text-only-documents. We discuss both of them in the following subsections.

4.2.1.1 Solutions for color documents

Existing methods to identify the source printer of color documents (i.e., documents with images) often exploit intrinsic signatures in the printing process, such as noise and geometric distortions, or in statistics derived from the transformed scanned images.

Solutions based on noise analysis Lee et al. [202, 203] used the CMYK color space to detect the printer source of a document. The authors calculate a residual image by subtracting the scanned version of a document to its Wiener-filtered version. The residual image is then summarized using gray-level co-occurence matrix (GLCM) statistics [204] and classified using a machine learning algorithm. Following a similar path, Choi et al. [205] and Tsai et al. [206] incorporated different color channels in the analysis and employed wavelets for feature extraction.

Elkasrawi and Shafait [200] also used the noise residual pattern to identify the printer even with common-resolution scans (400dpi). For this they propose a descriptor based on the work of Khanna et al. [207], in which statistics of the row and column directions of the image are calculated. However, image filtering is performed differently, with the aid of the Otsu's threshold method [208].

Solutions based on the analysis of geometric distortions Bulan et al. [209] used geometric distortions to identify the source of a color document. First, geometric signatures are extracted by estimating the positions of dots in halftone in training scanned documents of a given set of printers. Then, by correlation, the halftone points in a test document are linked to their source. Wu et al. [210] created printer models composed of distance and angles of halftone dots. K-means clustering on these Euclidean distances help in the final printer attribution process.

Solutions based on the analysis of statistics of the transformed image Ryu et al. [211] proposed the analysis of very high-resolution scanned images through histograms of Hough transform angles in CMYK color channels, generating a feature vector of printing patterns for each document printed by a given printer. The printer attribution is performed by correlating this pattern with a reference created for each printer.

Kim and Lee [212] used the halftone patterns for laser printer identification, acquiring images by photography, instead of scanning. First, the image is preprocessed to eliminate illumination variability using each channel in the CMY domain. Then, a set of 15 halftone texture features are extracted in the discrete Fourier transform domain and are used to feed a machine learning classifier. This work was extended upon in [196] using the curvelet transform and correlation-based attribution.

4.2.1.2 Solutions for text documents

For text documents, most of the approaches to printer attribution rely upon texture, noise and geometric distortion analysis in the printed letters to find the extrinsic signatures of the banding process common to different printers.

Solutions based on the texture of printed letters Mikkilineni et al. [193,213] proposed the use of texture descriptors based on statistics of gray-level co-occurrence matrices to identify the source of text documents. A set of letters "e", which is the most used letter in English texts [214], is chosen for feature extraction. Then, 22 statistics of gray-level co-occurrence matrices are extracted and used as input to a previously trained

5-nearest neighbors classifier, with the majority voting of the classified letters defining the final source of a document. In follow-up works, support vector machines (SVM) were used [197], as well as clustering and Euclidean distances [215]. Jiang et al. [216] proposed the extraction of feature vectors based on Benford's law. The extracted features were the first digit probability distribution of discrete cosine transform coefficients from multi-size blocks. Following a different path, Ali et al. [217] used the linearized pixel values of letters "i" as features properly mapped onto lower dimensional spaces through Principal Component Analysis. The decision making is then performed using a Gaussian mixture model machine learning classifier.

Ferreira et al. [195] proposed a series of approaches based on the multidirectionality and multiple resolution banding texture effects present in printed letters in a document. The authors extended the GLCM texture descriptor to consider more directions and scales in the analysis of the input letter. They also proposed another descriptor, called the convolutional texture gradient filter, which filters textures with specific gradient, present in areas that better differentiate the printers. The authors used the proposed approaches on "e" letters and proposed to consider another region for analysis: the frames, which are rectangular areas with sufficient printing material.

Finally, other authors have focused on the attribution problem for languages using different alphabets. Tsai et al. [198, 218] combined features from statistics of gray level co-occurrence matrices and sub-bands of wavelet transform for laser printer source of Chinese printed documents. As with English language, a specific symbol of Chinese language was chosen for analysis. Tsai et al. [199] extended upon this method by using statistical features from a gray-level co-occurrence matrix, discrete wavelet transform, spatial filter, Wiener filter and Gabor filter to identify the source of Japanese printed documents.

Solutions based on the analysis of noise and geometric distortions Kee and Farid [194] proposed to use reference characters and the reconstruction error to identify the source of text documents. The authors start with a reference "e" character of each printer. Then the search of similar characters from the same printer is done in a training step by template matching. These letters are then used to build the printer profile, useful for printer attribution later on. This profile is firstly built by preprocessing letters with histogram normalization and registration with respect to the reference letter of the printer. Then the mean character is calculated and the top p eigenvectors from principal component analysis [219] are calculated on aligned characters, yielding the printer profile.

Wu et al. [220] used geometric distortions to identify the laser printer source of documents. They first model a projective transformation using the center of characters and the whole scanned image in uncompressed format. Then, they solve this model with least squares and singular value decomposition for outlier removal. The estimated model parameters are used as geometric signatures inserted in a machine learning classifier. Finally, Schreyer [221] used statistical features from noise images in the discrete cosine transformed domain and in the multi-resolution wavelet domain, to train a machine learning classifiers for source printer attribution.

4.2.1.3 Remarks

In this work, instead of focusing on the printer attribution problem with hand-crafted features, similarly to previous solutions in the literature, we set ourselves the following guiding research principles:

1. Learn the discriminative features directly from the available collected data in a totally data-driven fashion.

2. Extract meaningful discriminative characteristics from a reduced set of training data, instead of the large ones often necessary for training deep Convolutional Neural Networks.

4.2.2 Proposed Method

The proposed solution for laser printer attribution works according to the following supervised machine learning pipeline. First, documents under analysis are digitalized and different sets of characters $\mathcal{S}_{\text{char}}$ are extracted from them (*e.g.,* $\mathcal{S}_{\text{e}}$ and $\mathcal{S}_{\text{a}}$ for characters "e" and "a", respectively). Each character of each set is processed separately. Characters are processed to obtain multiple representations of them (*i.e.,* $\mathcal{S}_{\text{char}}^{\text{raw}}$, $\mathcal{S}_{\text{char}}^{\text{med}}$ and $\mathcal{S}_{\text{char}}^{\text{avg}}$ contains the raw, median filtered residual and average filtered residual versions of the characters). For each representation, different features $\mathbf{f}_{\text{char}}^{\text{raw}}$, $\mathbf{f}_{\text{char}}^{\text{med}}$ and $\mathbf{f}_{\text{char}}^{\text{avg}}$ are extracted using small CNNs trained for this problem. These features are combined for each character set into a single feature vector $\mathbf{f}_{\text{char}}$, which is used to classify each character separately. Finally, a voting step aggregates all labels $l_{\text{char}}^{\text{print}}$ assigned to each character into a final decision l^{print}. In the following, we provide a detailed description of each step.

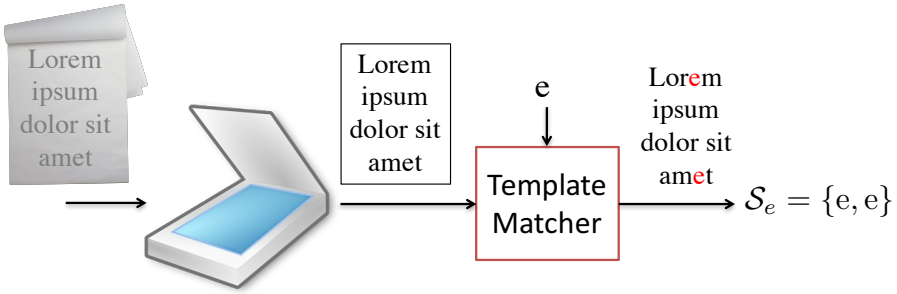


Figure 4.11: Document digitalization and character extractor pipeline. Printed documents are scanned, and letters are extracted by template matching using the procedure described in [195]. The set $\mathcal{S}_e$ is composed by the detected pixel patches containing character "e".

4.2.2.1 Characters extraction

Choosing the appropriate input data to solve laser printer attribution problem with the proposed architecture is an important step. As a matter of fact, selected data should contain enough information to characterize the used printer (e.g., banding artifacts). However, this data should not be strongly influenced by the semantic of the content, otherwise the network training would be negatively affected. As a good compromise, and motivated by state-of-the-art methods using characters as the minimal entity for text documents analysis [195], we also decided to start the analysis at character level.

To extract characters from printed documents, a digital version of them must be obtained. To this purpose, our algorithm starts by scanning all documents under analysis and extracting from scanned versions sets of characters using the same extractor devised by Ferreira et al. [195], as shown in Figure 4.11. The extractor works according to the following pipeline. First, we generate a reference letter, which has the same font typeface and is adjusted to have the same size as one letter from the scanned documents. Then, the algorithm slices the letter in eight regions and counts the proportion of black and white pixels in each one, yielding a feature vector used for letter extraction later on. To extract letters from the documents, black pixels connected components are extracted (i.e., character candidates) and the black/white ratio descriptor is computed again (the same as did before for the reference letter) for each connected component. Candidate letters whose descriptor has low cosine distance with respect to the reference letter descriptor are selected. Although the extractor is not perfect (the images of extracted letters have not the same size and some false positives may happen), it guarantees that most of the letters extracted are the same as the reference letter.

Figure 4.12: Same letter "e" printed by different printers.

4.2.2.2 Multiple representation of input data

By using different characters and different representations of them, it is possible to separately train several small networks in parallel instead of a single complex network, thus reducing the computational complexity and still achieving promising results. The intuition is that: (i) several simpler deep networks can be effectively trained using less training examples and (ii) early layers of simple networks are sufficient to identify interesting artifacts contained in the pixel domain (e.g., banding). Moreover, we also decided to consider different representations of the input data along with multiple simple deep networks. Different data representations rather than raw image pixels have already been considered in the forensic literature, such as for median filtering detection [45].

To this purpose, from each document, different sets $\mathcal{S}_{\text{char}}$ of grayscale characters of the same font and approximately the same size are extracted. As an example, a set $\mathcal{S}_{\text{e}}$ of letters "e" and $\mathcal{S}_{\text{a}}$ of letters "a" are used. In order to exploit the advantages given by multiple representation, for each set $\mathcal{S}_{\text{char}}$, we resorted to the following three different representations:

1. Raw data ($\mathcal{S}_{\text{char}}^{\text{raw}}$): image pixels are used as input to the network as they are. This is the common representation used as input for CNNs, as it contains high and low frequency components that can be isolated by the CNN filters and can be useful for image classification (see Figure 4.12).

2. Median filter residual ($\mathcal{S}_{\text{char}}^{\text{med}}$): we apply a 3×3 median filter over the image and subtract the image from the filtered version. The yielded noise pattern is used as input to the network. As the median filter better preserves edges, the median filter residual will contain, mostly, high frequency imperfections, which can be regarded as the banding (see Figure 4.13).

3. Average filter residual ($\mathcal{S}_{\text{char}}^{\text{avg}}$): we apply a 3×3 average filter over the image and subtract the image from its filtered version, using this residual as input to the network. This residual isolates border effects (see Figure 4.14).

Figure 4.13: Median filter residual representation of the same letters "e" showed in Figure 4.12. Here, some minimal borders are highlighted. Pixel values (black and white) are inverted in this figure for better visualization.

Figure 4.14: Average filter residual representation of the same letters "e" showed in Figure 4.12. Here, natural borders are highlighted. Pixel values (black and white) are inverted in this figure for better visualization.

4.2.2.3 Feature extraction

To extract relevant features from our input data, we use a deep learning approach. More specifically, we train a simple CNN for each character and each set $\mathcal{S}_{\text{char}}^{\text{raw}}$, $\mathcal{S}_{\text{char}}^{\text{med}}$ and $\mathcal{S}_{\text{char}}^{\text{avg}}$. Then we feed again patches from $\mathcal{S}_{\text{char}}^{\text{raw}}$, $\mathcal{S}_{\text{char}}^{\text{med}}$ and $\mathcal{S}_{\text{char}}^{\text{avg}}$ to the networks to obtain three feature vectors $\mathbf{f}_{\text{char}}^{\text{raw}}$, $\mathbf{f}_{\text{char}}^{\text{med}}$ and $\mathbf{f}_{\text{char}}^{\text{avg}}$ for each character within each set, using these vectors in a supervised classifier.

The used network architecture is common to each character and set and is similar in spirit to the MNIST network for digit recognition [222]. However, for a better representation of the data of interest herein, we train the network from scratch, yielding new filter weights able to recognize interesting characteristics for laser printer attribution. As far as we know, this is the first deep network custom-tailored to the printer attribution problem. The used CNN architecture has the following layers:

1. One input layer, where the raw image or a different representation (median filter residual or average filter residual) is used. It requires 28×28 images as input.

2. The first convolutional layer is made of 20 5×5 filters and is followed by a non-overlapping max pooling layer of size 2×2 and stride 2.

3. A second convolutional layer, with 50 filters of size $5 \times 5 \times 20$ is followed by another non-overlapping max pooling layer of size 2×2 and stride 2.

4. An inner product layer, which generates a vector $\in R^{500}$.

5. The 500 dimensional vector is non-linearly processed with a ReLU function applied element-wise.

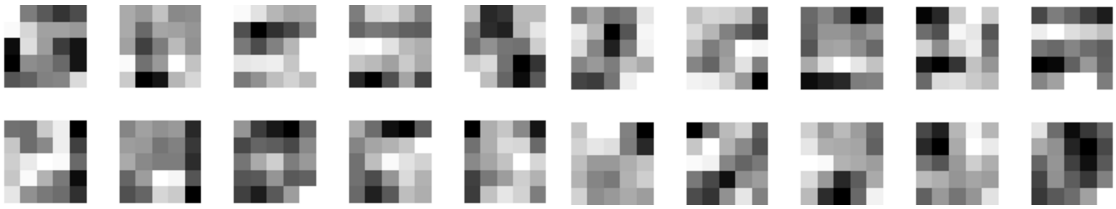


Figure 4.15: Example of filters weights for the first convolutional layer operating on the raw input image pixels. Weight values are mapped in grayscale.

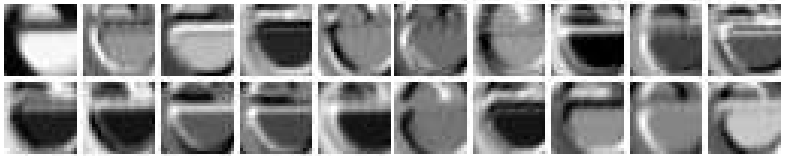


Figure 4.16: Convolutional output of the first layer of the trained network, given an input letter from an investigated printer. For each filter, different areas inside or outside the borders are highlighted.

6. An inner product layer acts as classifier with as many output confidence scores as the number of printers available during training.

7. A soft-max layer finally outputs the index and the confidence of the most probable printer.

In our proposed approach, we train the network using this architecture and then feed the training images again to the already trained the network, extracting 500-dimensional feature vectors in the last but one layer and repeating the process for the testing images. To follow the literature, we used the network as a feature extractor only, transferring the feature vectors to another and well used classifier for this application. The network autonomously learns which characteristics of the input images are relevant for discriminating the different printers.

Specifically, the network is trained using stochastic gradient descent with a momentum set to 0.9. We used an initial learning rate of 0.001 and a weight decay of 0.0005 without dropout. We used a batch size (subsampling of image examples used in one forward/backward pass through the network) of 100 images without batch normalization. The number of training epochs, which is the number of one forward and one backward pass of all training examples through the network was set to 30, and the model generated at the epoch with the smallest validation loss (20 epochs) was selected.

Figure 4.15 and Figure 4.16 depict the 20 5×5 filters of the first convolutional layer and also the characteristics they highlight from a letter printed by a given printer in the case the set $\mathcal{S}_{\mathrm{e}}^{\mathrm{raw}}$ is considered. These figures show that different filters enhance different

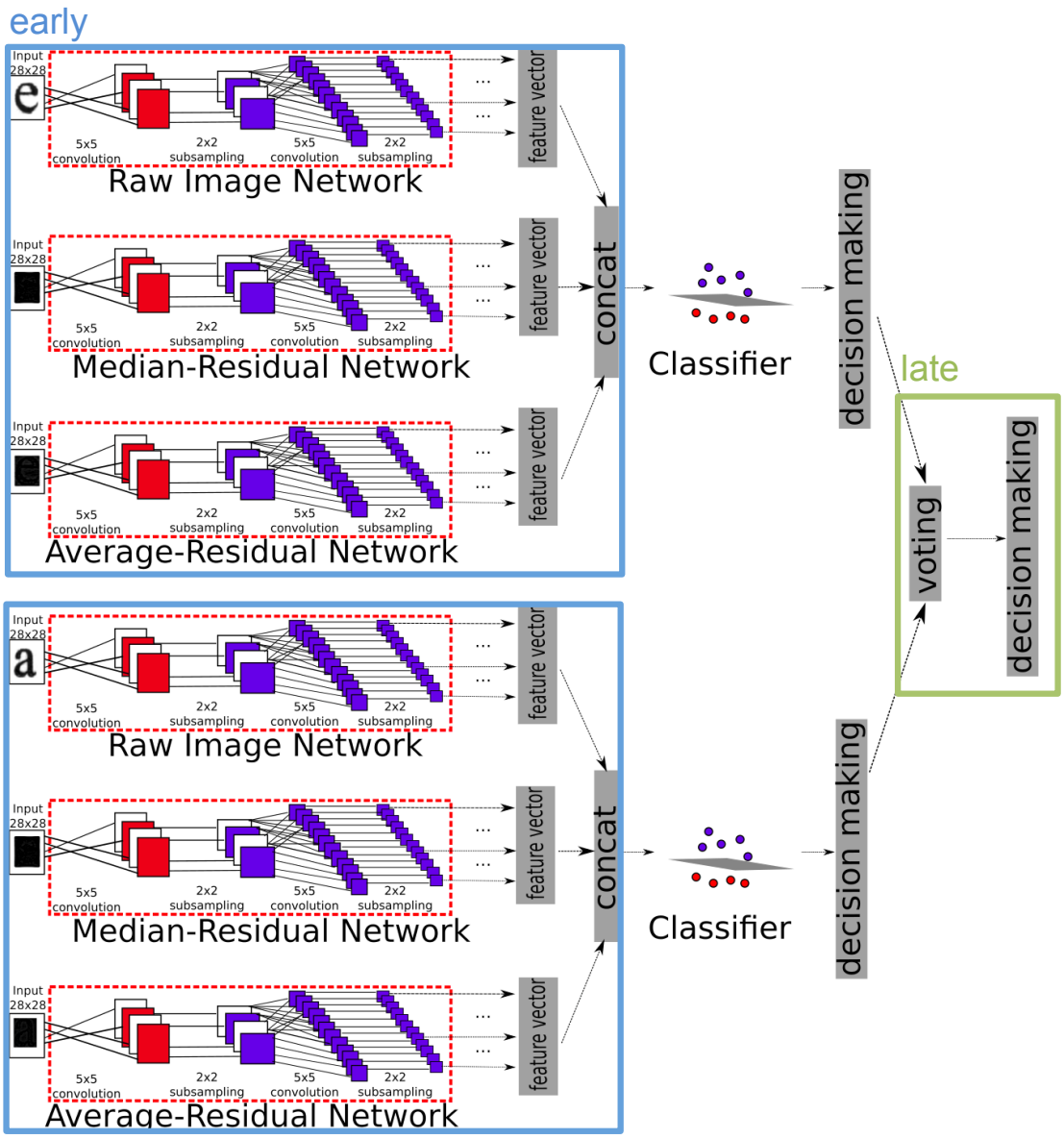


Figure 4.17: Proposed multiple representations of different data for laser printer attribution through a set of lightweight Convolutional Neural Networks. Early and late fusion steps are highlighted in blue and green, respectively.

areas of letters, such as texture and borders, which have been shown to be important to detect banding for LP attribution by existing methods in the literature such as [195].

4.2.2.4 Classification with early and late fusion

The proposed CNN architecture is characterized by a limited amount of parameters, in

order to allow a fast and reliable training even with a small number of labeled samples available. Small networks, as the one we are using, are expected to have worse performance with respect to bigger and deeper networks typically used in the computer vision community [37]. To compensate for this issue, we propose to use two lightweight fusion methods depicted in Figure 4.17:

1. **Early fusion – multiple representations of the same data:** we apply three different networks on input characters (of one type) coming from $\mathcal{S}_{\text{char}}^{\text{raw}}$, $\mathcal{S}_{\text{char}}^{\text{med}}$ and $\mathcal{S}_{\text{char}}^{\text{avg}}$. We concatenate the generated feature vectors $\mathbf{f}_{\text{char}}^{\text{raw}}$, $\mathbf{f}_{\text{char}}^{\text{med}}$ and $\mathbf{f}_{\text{char}}^{\text{avg}}$ into a single vector $\mathbf{f}_{\text{char}}$ in an early-fusion fashion [223]. This vector is fed to a set of linear SVMs used with a One-vs-One classification policy [224] to classify each character separately assigning a label $l_{\text{char}}^{\text{print}}$ to each one of them. The rationale for using this technique is that different representations highlight complementary artifacts.

2. **Late fusion – multiple representations of different data:** after taking decisions at the character level within a document, we apply a late-fusion technique [223] by using majority voting on sets of different characters. This is useful especially when dealing with documents presenting a limited amount of characters within a single set (e.g., only a few "e" letters). The obtained document label l^{print} allows us to pinpoint which printer was used to print the document.

For final decision making, we analyze the list of classification outcomes (votes) of letters from a document. In the case of ties, we decide the mode as being the first most frequent value that appears in the list. For example, in a list of classifications $x = [9, 7, 7, 7, 9, 9]$, the final classification would be 7. This can be thought of a pseudo-random tie-breaking and its most important advantage is simplicity. A more interesting tie-breaking policy would be be summing up the distances to the hyperplane of each classified letter in a document, per class, and then deciding the final class as being the one with the highest sum (i.e., surer about the classification from the classifier).

4.2.2.5 Remarks

As we show later on, it is indeed possible to train effective deep learning networks (DNNs) with less data if we take appropriate actions such as: (i) not selecting a too deep network, (ii) learning the features on different, and complementary, representations of the data; and (iii) combining the different outputs in the end through fusion. That being said, our motivation for using a solution based on a DNN for feature extraction

and a discriminative classifier at the end was threefold. First, we wanted to evaluate the richness of data-driven features directly and not the DNN as a full-fledged feature extractor + classifier. Although it is straightforward to attach a last soft-max layer in the end of the network for classification, we opted to use a discriminative classifier at the end to have a standardized form of comparison with previous works, which have used SVMs for classification. By doing this, we ended up having just one free comparison parameter (the features themselves). Second, our own previous experience with DNNs show that the combination of a DNN for feature extraction and a discriminative classifier at the end are very powerful, especially if we intend to perform fusion later on. Finally, our third motivation comes from that fact that by using a discriminative classifier at the end of the DNN-based feature extraction, we could simplify the fusion of different methods at the end, thus creating a lightweight integrated solution.

4.2.3 Experimental setup

This section presents the experimental methodology used in this work along with the used evaluation metrics, dataset and statistical tests. Finally, it details all the tested algorithms, some of which are baseline methods whereas some others are individual parts of our algorithm used to separately validate each step.

4.2.3.1 Dataset

For validation, we considered the same dataset of documents proposed by Ferreira et al. [195]. It comprises 120 Wikipedia documents containing up to three pages each converted to Portable Document Format (PDF). These documents were printed by 10 printers using $75g/m^2$ letter paper and scanned using a 600 dpi resolution Plustek SO PL2546 device, generating a total of 1,184 images. Table 4.5 shows the printers breakdown along with their main characteristics. This is the first standardized dataset in the literature containing documents in two languages: English and Portuguese. Although the characters in these two languages appear to be similar, in Portuguese texts, there are some accentuation signals in some letters (e.g., é and ã) that can confuse the letter extraction or the classification.

In [195], the authors have used two different datasets, one considering regions of interest of 980×800 pixels extracted from the input documents – referred to as `Frames Dataset` – and another one with only detected and extracted characters from the input documents – referred to as `Letters/Characters Dataset`. Given the results presented in [195], we further motivated our research to cope with the following real-world setups: (i) classifying documents for which only a few printed lines are available, making it impossible to extract

Table 4.5: Printers and number of documents per printer used in the dataset of Ferreira et al. [195]

ID	Brand	Model	Documents
B4070	Brother	HL-4070CDW	120
C1150	Canon	D1150	116
C3240	Canon	MF3240	120
C4370	Canon	MF4370DN	120
H1518	Hewlett Packard	CP1518	120
H225A	Hewlett Packard	CP2025	119
H225B	Hewlett Packard	CP2025	110
LE260	Lexmark	E260DN	119
OC330	OKI Data	C330DN	120
SC315	Samsung	CLP315	120
		Total	**1,184**

many frames and end up with a reliable attribution solution in an investigation; and (ii) having available only small pieces of document, a torn apart document or a shredded one. Those cases would render the analysis of frames impossible or useless.

Based on this motivation, we set forth the objective of tailoring a solution to the problem that would allow us to have the highest possible attribution effectiveness while, at the same time, not requiring large input regions from the investigated document. Thus, we decided to use the `Letters/Characters Dataset` presented in [195] as our reference benchmark. In addition, we also established the objective of exploring data-driven features directly learned from the data instead of hand-crafted oriented solutions as the ones exploited and reported in [195]. For that, we would need inputs that would not lead to an explosion of parameters in our DNN-oriented solution.

In addition to only using the "e" letters extracted from the documents as [195] we go beyond and exploit the impact of using different letters as well, as the authors in [195] did not consider these cases. Table 4.6 summarizes the datasets of letters used for the tests we generated from the aforementioned documents. As already mentioned, these have been extracted exploiting the characters extractor devised by Ferreira et al. [195]. With this method, we extracted several different letters of approximately 38×47 pixels printed with the Wikipedia font from the documents according to their frequency in the English language [214], resulting in four datasets of different letters $\mathbb{D}_{\mathrm{e}}$, $\mathbb{D}_{\mathrm{a}}$, $\mathbb{D}_{\mathrm{d}}$, $\mathbb{D}_{\mathrm{o}}$ as reported in Table 4.6.

As the extractor in [195] also detects letters of similar font and size, these datasets can be regarded as affected by a small amount of noise. As an example, Figure 4.18 shows the distribution of sizes of the extracted "e" letters for each printer. Although most of them share a common size, some of them slightly deviate. These datasets are then very useful to test the performance of the proposed algorithm in adverse conditions.

To validate the proposed method in a noiseless scenario, we also created a clean dataset $\tilde{\mathbb{D}}_{\mathrm{e}}$ of $131,435$ "e" letters. This dataset was created starting from the noisy "e" dataset

Table 4.6: Datasets used for experimental evaluation.

Dataset	Letter	Samples
$\mathbb{D}_{\text{e}}$	"e"	245,650
$\mathbb{D}_{\text{a}}$	"a"	286,098
$\mathbb{D}_{\text{d}}$	"d"	185,009
$\mathbb{D}_{\text{o}}$	"o"	351,850
$\tilde{\mathbb{D}}_{\text{e}}$	"e"	131,435
$\mathbb{D}_{\text{frame}}$	frames	352,433

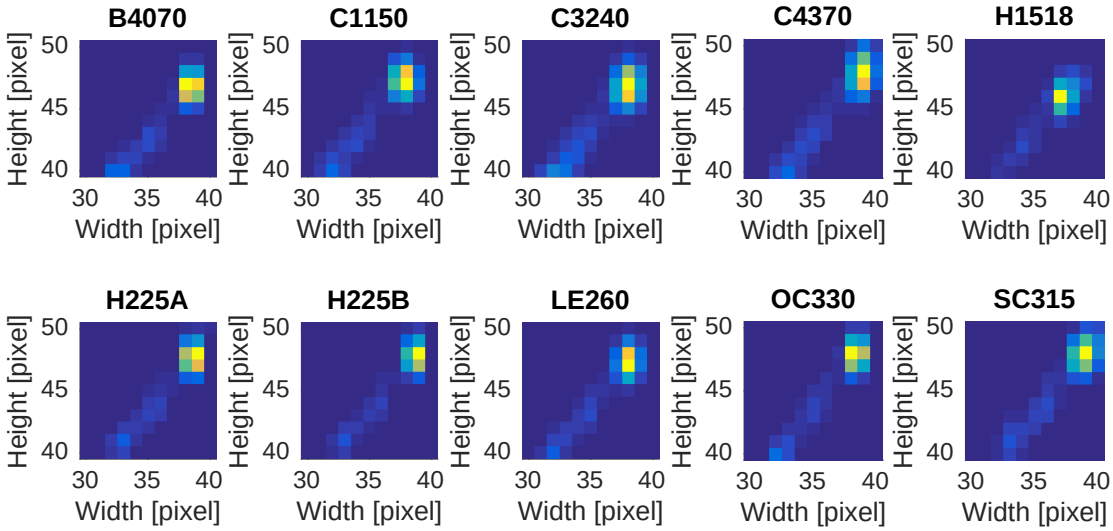


Figure 4.18: Distribution of the extracted letter "e" sizes for each printer. Most of the characters have a resolution of 38×47 pixels, but some have slightly different sizes.

$\mathbb{D}_{\text{e}}$ keeping only the most similar letters sharing a $(38 \pm 1) \times (47 \pm 1)$ resolution.

At this point, it is worth mentioning that the input of our network is always a 28×28 pixel patch. Therefore, we always crop the center region of the letters so as to have inputs exactly matching this network requirement. We do not perform any resampling/resizing in order to avoid introducing additional processing artifacts that can hinder attribution performances by masking part of the telltales left behind by printers.

Finally, to validate our idea of using characters to train our small CNNs, we also built a dataset of small frames (e.g., small random patches). To this purpose, we applied a 28×28 frame extractor in the documents, extracting 300 valid frames whose ratio between black and white pixels r is $0.6 \leq r \leq 0.8$ from each scanned document. This resulted in the $\mathbb{D}_{\text{frame}}$ dataset.

4.2.3.2 Experimental methodology

For validation, we consider the same 5×2 cross-validation protocol used in [195]. In this protocol, we replicate the traditional 2-fold cross-validation protocol five times (thus 5×2). In each of these 2-fold cross validations, a set of documents (not characters) $\mathbb{D}$ is split into $\mathbb{D}_1$ and $\mathbb{D}_2$. In each of the five executions, a classifier is trained with characters of documents in $\mathbb{D}_1$ and tested on characters present in $\mathbb{D}_2$, and then vice-versa. After that, we report the results based on documents classification (after majority voting of test documents letters) and perform the statistical tests after 10 rounds of experiments. In this experimental protocol, each combination of training and test will use letters from 592 documents for training an one-against-one SVM classifier while the remaining 592 documents letters are used for testing the classifier. The number of letters used in the training and testing of each of 10 experiments (which we call fold) will depend on how many letters are extracted from each training and testing document and will also depend on which letter is being used in the analysis. For example, in the total 5×2 protocol, there are a mean of 122,825 letters 'e' for training and the same for testing. According to a study conducted by Dietterich et al. [225], the 5×2 cross-validation is considered an optimal experimental protocol for learning algorithms.

In a multi-class problem with n classes, the classification results may be represented in an $n \times n$ confusion matrix. In this case, the main diagonal contains the correct classifications while the other entries contain misclassifications. In the 5×2 cross validation protocol, one confusion matrix is yielded per experiment. Therefore, we present results by averaging these matrices.

To test the statistical relevance of the obtained experimental results, we consider a two-level statistical test. In the first level, we use the Friedman test as a pre-test to point out whether or not there is statistical difference in the obtained results. Then we refine these results with the Tukey-Kramer post-test, also known as honestly significant difference (HSD) test to point out statistical differences (if any) pairwise. In all tests, we set the confidence level to 95%.

4.2.3.3 Tested algorithms

We performed several tests to validate the proposed approach. First, we conducted a set of experiments aimed at selecting the reference CNN architecture. Then we tested each separate step of our algorithm (e.g., robustness to noise, early fusion, late fusion, etc.). Finally, we validated the proposed algorithm against state-of-the-art baseline methods.

At first we compared several different Convolutional Neural Network architectures in

order to find the right balance between complexity and accuracy. To this purpose, in addition to the architecture proposed in Section 4.2.2, hereinafter denoted as $\mathcal{S}^{2\text{-}Conv}$, we also tested some deeper solutions. By adding one and two more convolutional layers, each followed by a max-pooling layer, we created two CNNs, denoted as $\mathcal{S}^{3\text{-}Conv}$ and $\mathcal{S}^{4\text{-}Conv}$. Two additional state of the art network architectures were used as benchmark: AlexNet [35], denoted as $S^{AlexNet}$, and GoogLeNet [37], denoted as $S^{GoogLeNet}$.

After validating the use of $\mathcal{S}^{2\text{-}Conv}$ as CNN (hereinafter simply denoted as $\mathcal{S}$ for the sake of clarity), we also tested each data representation separately. This means we extracted features using CNNs on a single representation of the input data (e.g., raw letters "e") and used the obtained feature vectors for classification with SVM. Majority voting was applied to letters to take a decision at document level. As single representations, we tested the median filter residual of the image ($\mathcal{S}^{\text{med}}_{\text{char}}$), the average filter residual ($\mathcal{S}^{\text{avg}}_{\text{char}}$) and the raw image pixels ($\mathcal{S}^{\text{raw}}_{\text{char}}$).We also tested different representations inspired by the existing methods in the literature. As a matter of fact, we tested the filtered image from the Convolutional Texture Gradient Filter (CTGF) using both the 3×3 ($\mathcal{S}^{\text{CTGF3}}_{\text{char}}$) and the 5×5 ($\mathcal{S}^{\text{CTGF5}}_{\text{char}}$) filters from the work of Ferreira et al. [195] and also the Wiener filter residual [226] ($\mathcal{S}^{\text{Wiener}}_{\text{char}}$).For each approach, the subscript "char" represents the letter we tested (e.g., $\mathcal{S}^{\text{Wiener}}_{\text{e}}$ for the Wiener-based representation on "e" letters). With an abuse of notation, we use the symbol $\mathcal{S}$ to refer to both the algorithm and the set of input data.

We also tested the performance of the early fusion approach. For this, we concatenated the feature vectors from the last but one layer of CNNs applied on three different representations of the same data, making them the input of an SVM classifier.We refer to early fusion methods as $\{\mathcal{S}^{\text{raw}}, \mathcal{S}^{\text{med}}, \mathcal{S}^{\text{avg}}\}_{\text{char}}$, where the methods in the brackets represent the used data representations, and the subscript indicates the used letter (i.e., "e", "a", "d", "o" or 28×28 frames).

To test the late fusion, we performed majority voting to classification labels obtained with early fusion methods run on different character families. We call these approaches $\{\mathcal{S}^{\text{raw}}, \mathcal{S}^{\text{med}}, \mathcal{S}^{\text{avg}}\}_{\text{char1, ..., charN}}$, specifying the different sets of characters used for fusion. Notice that late fusion approaches also embed early fusion.

We also compared our proposed technique to eight state-of-the-art methods focused on text documents. The first one is the GLCM-based method from Mikkilineni et al. [193, 213], which describes the signature present in the banding with 22 statistics calculated per matrix. We refer to this approach in the experiments `GLCM`. The next four methods used in the experiments were proposed in the work of Ferreira et al. [195]. The first one uses GLCM with multiple directions (`GLCM-MD`), while the second uses GLCM with multiple directions and multiple scales in the input data (`GLCM-MD-MS`). The third one

uses CTGF with size 3×3 (`CTGF`) and the fourth method uses a combination of all these methods (`CTGF-GLCM-MD-MS`).

The sixth implemented method from the literature was proposed by Kee and Farid [194] (`RECONST-ERR`) and uses reference characters to extract letters from documents. To detect the source of a document, letters "e" are extracted and compared with the profile of each printer to obtain a reconstruction error for each printer. The printer with the smallest mean error is detected as the source. Finally, we also tested two well-known texture descriptors widely used in the literature: (i) local binary patterns (`LBP`) [227]; and (ii) histogram of oriented gradients [228] (`HOG`).

4.2.4 Results and discussion

We now turn our attention to the experimental results obtained with different methods. First, we test our proposed lightweight CNN fusion approach against several individual CNN architectures. Second, we dissect the proposed approach to test each of its steps separately. Third, we show results considering the effects of training CNNs on noisy rather than noiseless data. Then, we compare different representations of the input data. Afterwards, we show the advantages of using multiple representations (early fusion) and multiple data (late fusion). Finally, we present experiments comparing the performance of our approach to the methods discussed in Section 4.2.3.3. All experiments were performed using the methodology presented in Section 4.2.3.2 on the dataset with 1,184 printings presented in Section 4.2.3.1.

4.2.4.1 Evaluation of the CNN model

The first step toward the development of our proposed deep learning approach for laser printer attribution is to determine the kind of CNN architecture that best suits the problem at hand. One natural solution would be using the whole digitalized document as input for a Convolutional Neural Network, but this procedure have the following drawbacks: (i) it requires the designing of deeper networks, which will require a larger amount of data, computational time and memory resources to train the network; and (ii) the network training process will be strongly influenced by the semantic of the documents. Conversely, smaller areas with fixed patterns used as input to smaller networks do not require as many layers as using the whole document as input and also can lead to a faster learning of network parameters and weights.

In this vein, we selected CNNs whose input are small patches of size as 28×28, 227×227 and 224×224 as candidate architectures for our proposed multiple represen-

Table 4.7: Results comparing different deep learning approaches for laser printer attribution in one combination of training and testing. Our best proposed late fusion approach is highlighted in light gray. TTE refers to the training time for a single epoch.

Method	Accuracy	TTE [s]	Size [MB]	Input Data
$\{\mathcal{S}^{\text{raw}}, \mathcal{S}^{\text{med}}, \mathcal{S}^{\text{avg}}\}_{\text{a,e}}$	98.30%	20.22	9.84	$\mathbb{D}_{\text{a}}, \mathbb{D}_{\text{e}}$
$\mathcal{S}^{GoogLeNet}$ [37]	98.30%	886.00	39.40	$\mathbb{D}_{\text{e}}$
$\mathcal{S}^{AlexNet}$ [35]	98.13%	290.00	217.00	$\mathbb{D}_{\text{e}}$
$S_e^{4\text{-}Conv,raw}$	97.29%	21.70	15.43	$\mathbb{D}_{\text{e}}$
$S_e^{3\text{-}Conv,raw}$	96.10%	8.10	2.92	$\mathbb{D}_{\text{e}}$

tation of multiple data approach. For each candidate architecture, we train and test the first split of the raw "e" dataset $\mathbb{D}_{\text{e}}$, training these architectures for 30 epochs. The model generated at the epoch with the smallest validation loss is selected as the best candidate for each CNN. We show in Table 4.7 results considering our fusion approach, denoted as $\{\mathcal{S}^{\text{raw}}, \mathcal{S}^{\text{med}}, \mathcal{S}^{\text{avg}}\}_{\text{a,e}}$, using six lightweight networks with 2 convolutional layers (architecture that we denote as $\mathcal{S}^{2\text{-}Conv}$) and some individual deeper architectures, using the networks as feature extractors and a linear SVM as the classifier.

As shown in Table 4.7, the proposed approach, underpinned by six lightweight networks instead of one, has similar results to a more complex network ($\mathcal{S}^{GoogLeNet}$) while presenting a memory footprint 75% more efficient and being, approximately, 43× faster to train. Moreover, each individual network of the proposed late-fusion approach consumes 1.64MB, thus the final footprint is $6 \times 1.64 = 9.84$MB of space. This is a further confirmation that the use of the proposed lightweight simple networks in a fusion framework outperforms deeper solutions in terms of complexity-accuracy trade-off, at least for the particular setup considered herein. Indeed, the fusion approach with six networks reaches an accuracy equals the one generated by a deeper network, but with a reduced complexity.

We also evaluate the solutions for laser printer attribution with different training set sizes. We start comparing our proposed lightweight fusion of CNNs to existing solutions for laser printer attribution on different proportions of training data. For this experiment, we separated one combination of training and test data, sub-sampling the training data to be 1%, 10%, 30%, 50%, 70%, and 100% of the original training data, classifying the same testing data using the same SVM linear classifier used in the experiments. We show the results in Table 4.8. Each column shows a percentage of the training data used.

Normally, deeper networks require more data to show good results if compared to smaller ones. As Table 4.8 shows, the proposed approach outperforms more complex

Table 4.8: Results comparing different deep learning approaches against our proposed approach for laser printer attribution in one combination of training and testing on different amounts of training data. The best accuracy per training data proportion used is highlighted in gray.

Method	1%	10%	30%	50%	70%	100%
$\{\mathcal{S}^{\text{raw}}, \mathcal{S}^{\text{med}}, \mathcal{S}^{\text{avg}}\}_{\text{a,e}}$	91.20%	97.29%	97.63%	97.63%	97.96%	98.30%
$\mathcal{S}^{GoogLeNet}$ [37]	87.64%	96.27%	94.92%	94.07%	98.13%	98.30%
$\mathcal{S}^{AlexNet}$ [35]	89.00%	96.44%	95.77%	96.44%	97.29%	98.13%

CNNs for 1%, 10%, 30% and 50% of training data proportion used. A deeper network (GoogLeNet) starts to catch up and outperforms the proposed method when using 70% of data. In summary, the fusion of $\mathcal{S}^{2\text{-}Conv}$ networks has the following advantages (i) it requires less data for effective training; and (ii) individually, each network (S_2 architecture) used in the fusion requires less memory and time to train than using more complex networks. Therefore, we chose $\mathcal{S}^{2\text{-}Conv}$ architecture in our proposed fusion approach. In the following, we will denote the $\mathcal{S}^{2\text{-}Conv}$ architecture simply as $\mathcal{S}$ in order to allow for a more compact notation.

4.2.4.2 Dealing with noisy data

In order to be useful in a real-world scenario, it is important that the developed method is robust against non-ideal working conditions. More specifically, it is paramount that the features learned by the CNNs are generalizable enough to guarantee good performance also on noisy data (e.g., letters of slightly different sizes). To test this property, we trained and tested the algorithm using different single representations of the "e" character (i.e., $\mathcal{S}_{\text{e}}^{\text{raw}}$, $\mathcal{S}_{\text{e}}^{\text{avg}}$, and $\mathcal{S}_{\text{e}}^{\text{med}}$) on different combinations of datasets (i.e., the noiseless $\tilde{\mathbb{D}}_{\text{e}}$ and the noisy $\mathbb{D}_{\text{e}}$).

Table 4.9 shows the achieved results. For each representation, the best accuracy (around 97%) is obtained when the algorithm is trained and tested on clean data not containing characters at different size ($\tilde{\mathbb{D}}_{\text{e}}$). When the same network trained on clean data ($\tilde{\mathbb{D}}_{\text{e}}$) is tested against dirty data ($\mathbb{D}_{\text{e}}$), accuracy falls down at approximately 85%. However, it is sufficient to train CNNs on $\mathbb{D}_{\text{e}}$ to obtain results comparable to the noiseless case even when dirty data is tested (94%). Therefore, to ensure enough robustness, from this point on, we always consider noisy datasets for both training and testing, as they are closer to a real-world setup.

Table 4.9: Average results using early fusion and single representations on noiseless ($\tilde{\mathbb{D}}_{\mathrm{e}}$) and noisy ($\mathbb{D}_{\mathrm{e}}$) datasets.

Method	Mean	Training Data	Test Data
$\mathcal{S}_{\mathrm{e}}^{\mathrm{raw}}$	97.95%	$\tilde{\mathbb{D}}_{\mathrm{e}}$	$\tilde{\mathbb{D}}_{\mathrm{e}}$
	96.13%	$\mathbb{D}_{\mathrm{e}}$	$\mathbb{D}_{\mathrm{e}}$
	84.43%	$\tilde{\mathbb{D}}_{\mathrm{e}}$	$\mathbb{D}_{\mathrm{e}}$
$\mathcal{S}_{\mathrm{e}}^{\mathrm{avg}}$	97.56%	$\tilde{\mathbb{D}}_{\mathrm{e}}$	$\tilde{\mathbb{D}}_{\mathrm{e}}$
	94.50%	$\mathbb{D}_{\mathrm{e}}$	$\mathbb{D}_{\mathrm{e}}$
	85.81%	$\tilde{\mathbb{D}}_{\mathrm{e}}$	$\mathbb{D}_{\mathrm{e}}$
$\mathcal{S}_{\mathrm{e}}^{\mathrm{med}}$	96.87%	$\tilde{\mathbb{D}}_{\mathrm{e}}$	$\tilde{\mathbb{D}}_{\mathrm{e}}$
	94.30%	$\mathbb{D}_{\mathrm{e}}$	$\mathbb{D}_{\mathrm{e}}$
	85.58%	$\tilde{\mathbb{D}}_{\mathrm{e}}$	$\mathbb{D}_{\mathrm{e}}$

Table 4.10: Results obtained using different representations on different datasets sorted from best to worst.

Method	Mean ± Std.Dev.	Input Data
$\mathcal{S}_{\mathrm{e}}^{\mathrm{raw}}$	96.13% ± 0.00	$\mathbb{D}_{\mathrm{e}}$
$\mathcal{S}_{\mathrm{a}}^{\mathrm{avg}}$	94.89% ± 0.30	$\mathbb{D}_{\mathrm{a}}$
$\mathcal{S}_{\mathrm{e}}^{\mathrm{avg}}$	94.50% ± 0.03	$\mathbb{D}_{\mathrm{e}}$
$\mathcal{S}_{\mathrm{e}}^{\mathrm{med}}$	94.30% ± 0.01	$\mathbb{D}_{\mathrm{e}}$
$\mathcal{S}_{\mathrm{a}}^{\mathrm{med}}$	93.34% ± 0.02	$\mathbb{D}_{\mathrm{a}}$
$\mathcal{S}_{\mathrm{a}}^{\mathrm{raw}}$	93.07% ± 0.03	$\mathbb{D}_{\mathrm{a}}$
$\mathcal{S}_{\mathrm{e}}^{\mathrm{CTGF3}}$	89.12% ± 0.03	$\mathbb{D}_{\mathrm{e}}$
$\mathcal{S}_{\mathrm{e}}^{\mathrm{Wiener}}$	84.84% ± 0.30	$\mathbb{D}_{\mathrm{e}}$
$\mathcal{S}_{\mathrm{e}}^{\mathrm{CTGF5}}$	83.15% ± 0.06	$\mathbb{D}_{\mathrm{e}}$

4.2.4.3 Choice of multiple representations

The proposed algorithm works by exploiting multiple representations of the input data. It is therefore important to detect which representations contain more discriminative information for LP attribution. Table 4.10 shows the best results obtained using different representations (e.g., $\mathcal{S}_{\mathrm{char}}^{\mathrm{raw}}$, $\mathcal{S}_{\mathrm{char}}^{\mathrm{Wiener}}$, etc.) on the different datasets (e.g., $\mathbb{D}_{\mathrm{a}}$, $\mathbb{D}_{\mathrm{e}}$, $\mathbb{D}_{\mathrm{o}}$, etc.).

Representations yielding higher accuracies are $\mathcal{S}_{\mathrm{char}}^{\mathrm{raw}}$, $\mathcal{S}_{\mathrm{char}}^{\mathrm{avg}}$ and $\mathcal{S}_{\mathrm{char}}^{\mathrm{med}}$, whereas the use of CTGF or Wiener-filtered versions of the characters provide the worst results. The best results are obtained using "a" and "e" datasets. This can be explained as they are the most common characters in English and Portuguese. Therefore, $\mathbb{D}_{\mathrm{a}}$ and $\mathbb{D}_{\mathrm{e}}$ are larger than $\mathbb{D}_{\mathrm{d}}$, whereas $\mathbb{D}_{\mathrm{o}}$ probably is affected by too much noise as "o" can be often mistaken with other letters during the characters extraction phase.

Interestingly, for some data (letters), the raw representation in deep networks is not good enough. For instance, deep networks applied on average filter residual ($\mathcal{S}_{\mathrm{a}}^{\mathrm{avg}}$) of letters "a" yielded an accuracy of 94.89%, against the accuracy of 93.07% on letters "a"

Table 4.11: Results comparing early and late fusion using the best representations sorted from best to worst. Late fusion approaches are highlighted in light gray.

Method	Mean ± Std.Dev.	Input Data
$\{\mathcal{S}^{\text{raw}}, \mathcal{S}^{\text{med}}, \mathcal{S}^{\text{avg}}\}_{\text{a,e}}$	97.33% ± 0.00	$\mathbb{D}_{\text{a}}, \mathbb{D}_{\text{e}}$
$\{\mathcal{S}^{\text{raw}}, \mathcal{S}^{\text{med}}, \mathcal{S}^{\text{avg}}\}_{\text{a}}$	96.89% ± 0.00	$\mathbb{D}_{\text{a}}$
$\{\mathcal{S}^{\text{raw}}, \mathcal{S}^{\text{med}}, \mathcal{S}^{\text{avg}}\}_{\text{a,e,d}}$	96.87% ± 0.00	$\mathbb{D}_{\text{a}}, \mathbb{D}_{\text{e}}, \mathbb{D}_{\text{d}}$
$\{\mathcal{S}^{\text{raw}}, \mathcal{S}^{\text{med}}, \mathcal{S}^{\text{avg}}\}_{\text{e}}$	96.84% ± 0.00	$\mathbb{D}_{\text{e}}$
$\{\mathcal{S}^{\text{raw}}, \mathcal{S}^{\text{med}}, \mathcal{S}^{\text{avg}}\}_{\text{a,e,o}}$	96.24% ± 0.03	$\mathbb{D}_{\text{a}}, \mathbb{D}_{\text{e}}, \mathbb{D}_{\text{o}}$
$\{\mathcal{S}^{\text{raw}}, \mathcal{S}^{\text{med}}, \mathcal{S}^{\text{avg}}\}_{\text{d}}$	93.67% ± 0.03	$\mathbb{D}_{\text{d}}$
$\{\mathcal{S}^{\text{raw}}, \mathcal{S}^{\text{med}}, \mathcal{S}^{\text{avg}}\}_{\text{o}}$	92.21% ± 0.03	$\mathbb{D}_{\text{o}}$
$\{\mathcal{S}^{\text{raw}}, \mathcal{S}^{\text{med}}, \mathcal{S}^{\text{avg}}\}_{\text{a,e,frame}}$	88.72% ± 0.02	$\mathbb{D}_{\text{a}}, \mathbb{D}_{\text{e}}, \mathbb{D}_{\text{frames}}$
$\{\mathcal{S}^{\text{raw}}, \mathcal{S}^{\text{med}}, \mathcal{S}^{\text{avg}}\}_{\text{frame}}$	73.69% ± 0.05	$\mathbb{D}_{\text{frames}}$

raw image pixels ($\mathcal{S}^{\text{raw}}_{\text{a}}$). This justifies the use of multiple representations and motivates the use of data fusion.

4.2.4.4 Early and late fusion

To validate the early and late fusion stages, we tested only the three selected best representations $\mathcal{S}^{\text{raw}}_{\text{char}}$, $\mathcal{S}^{\text{avg}}_{\text{char}}$ and $\mathcal{S}^{\text{med}}_{\text{char}}$. Table 4.11 shows the results of the 5×2 cross-validation experiments considering this scenario.

Fusion approaches typically outperform the ones using only single representations. This is because different representations in the input layers of CNNs can contain important information that better identifies the banding over the different networks, as well as other printing artifacts left behind during the physical printing of a document. For example, banding in the borders contained in the average filter residual are better highlighted in its CNN and can complement the information found in the two other CNNs that use information from the raw image data and median filter residual. Moreover, different letters (late fusion) can contain even more explicit banding patterns than using the same letter. With these findings, we conclude that both multiple representation approach and late-fusion are useful for laser printer attribution using deep networks.

A special comment is in order regarding the use of frames ($\{\mathcal{S}^{\text{raw}}, \mathcal{S}^{\text{med}}, \mathcal{S}^{\text{avg}}\}_{\text{frame}}$). As a matter of fact, their use, instead of letters, is not as effective when deploying a solution using deep learning. This is explained by the fact that different data are used as input at the same time to the same network, each of them presenting different printing patterns, probably demanding a different and deeper CNN architecture. This further confirms the idea of using characters for the proposed method.

Considering all the presented results, the statistical test using the Friedmann pretest yielded the p-value of 7.55×10^{-154}, helping us to state that the approaches have a statistical significant difference. Table 4.12 shows the statistical Tukey HSD tests. This confirms that our proposed fusion approaches have statistically significant difference

Table 4.12: Tukey-HSD pairwise statistical tests considering CNN approaches that use unique and multiple data.

Rank	Method	$\{S^{raw}, S^{med}, S^{avg}\}_{a,e}$	$\{S^{raw}, S^{med}, S^{avg}\}_{a}$	$\{S^{raw}, S^{med}, S^{avg}\}_{a,e,d}$	$\{S^{raw}, S^{med}, S^{avg}\}_{e}$	$\{S^{raw}, S^{med}, S^{avg}\}_{a,e,o}$	S^{raw}_{e}	S^{avg}_{a}	S^{avg}_{e}	S^{med}_{e}	$\{S^{raw}, S^{med}, S^{avg}\}_{d}$	S^{med}_{a}	S^{raw}_{a}	$\{S^{raw}, S^{med}, S^{avg}\}_{o}$	S^{CTGF3}_{a}	$\{S^{raw}, S^{med}, S^{avg}\}_{a,e,frame}$	S^{wiener}_{e}	S^{CTGF5}_{e}	$\{S^{raw}, S^{med}, S^{avg}\}_{frame}$	TOTAL
1	$\{S^{raw}, S^{med}, S^{avg}\}_{a,e}$	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	14
2	$\{S^{raw}, S^{med}, S^{avg}\}_{a}$	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	12
3	$\{S^{raw}, S^{med}, S^{avg}\}_{a,e,d}$	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	14
4	$\{S^{raw}, S^{med}, S^{avg}\}_{e}$	0	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	11
5	$\{S^{raw}, S^{med}, S^{avg}\}_{a,e,o}$	-1	0	-1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	2
6	S^{raw}_{e}	-1	0	-1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	2
7	S^{avg}_{a}	-1	-1	-1	-1	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0
8	S^{avg}_{e}	-1	-1	-1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1
9	S^{med}_{e}	-1	-1	-1	-1	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0
10	$\{S^{raw}, S^{med}, S^{avg}\}_{d}$	-1	-1	-1	-1	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0
11	S^{med}_{a}	-1	-1	-1	-1	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0
12	S^{raw}_{a}	-1	-1	-1	-1	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0
13	$\{S^{raw}, S^{med}, S^{avg}\}_{o}$	-1	-1	-1	-1	0	0	0	0	0	0	0	0	0	1	0	0	1	1	-1
14	S^{CTGF3}_{a}	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	-1	0	0	-14
15	$\{S^{raw}, S^{med}, S^{avg}\}_{a,e,frame}$	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	0	0	0	0	-12
16	S^{wiener}_{e}	-1	-1	-1	-1	0	0	0	0	0	0	0	0	0	1	0	0	1	1	-1
17	S^{CTGF5}_{e}	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	-1	0	0	-14
18	$\{S^{raw}, S^{med}, S^{avg}\}_{frame}$	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	-1	0	0	-14

1 = Line method is better than column method
0 = Line method is equivalent to column method
-1 = Line method is worse than column method

when compared to all the single representations. Notice that, even if the results obtained using early and late fusion are statistically equivalent, the use of late fusion is strongly motivated whenever a document does not contain enough letters from the same set (e.g., enough "e" letters).

4.2.4.5 Comparison with existing techniques in the literature

Table 4.13 shows the results of the 5×2 cross-validation experiments considering our best

Table 4.13: Results comparing the best configurations of the proposed method to the existing methods in the literature after 5×2 validation. Late fusion approaches are highlighted in light gray.

Method	Mean ± Std.Dev.	Input Data
$\{\mathcal{S}^{\text{raw}}, \mathcal{S}^{\text{med}}, \mathcal{S}^{\text{avg}}\}_{\text{a,e}}$	97.33% ± 0.0065	$\mathbb{D}_{\text{a}}, \mathbb{D}_{\text{e}}$
$\{\mathcal{S}^{\text{raw}}, \mathcal{S}^{\text{med}}, \mathcal{S}^{\text{avg}}\}_{\text{a}}$	96.89% ± 0.0052	$\mathbb{D}_{\text{a}}$
$\{\mathcal{S}^{\text{raw}}, \mathcal{S}^{\text{med}}, \mathcal{S}^{\text{avg}}\}_{\text{a,e,d}}$	96.87% ± 0.0087	$\mathbb{D}_{\text{a}}, \mathbb{D}_{\text{e}}, \mathbb{D}_{\text{d}}$
$\{\mathcal{S}^{\text{raw}}, \mathcal{S}^{\text{med}}, \mathcal{S}^{\text{avg}}\}_{\text{e}}$	96.84% ± 0.0068	$\mathbb{D}_{\text{e}}$
`CTGF-GLCM-MD-MS` [195]	96.26% ± 0.0054	$\mathbb{D}_{\text{e}}$
$S_e^{4\text{-}Conv,raw}$	95.84% ± 1.4700	$\mathbb{D}_{\text{e}}$
$S_e^{3\text{-}Conv,raw}$	95.40% ± 0.8400	$\mathbb{D}_{\text{e}}$
`GLCM-MD-MS` [195]	94.30% ± 0.0110	$\mathbb{D}_{\text{e}}$
`GLCM-MD` [195]	91.08% ± 0.0089	$\mathbb{D}_{\text{e}}$
`HOG` [228]	90.59% ± 0.0214	$\mathbb{D}_{\text{e}}$
`LBP` [227]	88.66% ± 0.0145	$\mathbb{D}_{\text{e}}$
`RECONST-ERR` [194]	78.90% ± 0.0210	$\mathbb{D}_{\text{e}}$
`GLCM` [193, 213]	77.87% ± 0.0459	$\mathbb{D}_{\text{e}}$
`CTGF` [195]	72.46% ± 0.0377	$\mathbb{D}_{\text{e}}$

approaches and existing counterparts in the literature. In this scenario, we are using all approaches as feature extractors and feeding a linear SVM classifier with these vectors in the training and testing step.

Table 4.13 shows that the first proposed method that outperforms the state of the art is the one that uses multiple representations of the letter "e" ($\{\mathcal{S}^{\text{raw}}, \mathcal{S}^{\text{med}}, \mathcal{S}^{\text{avg}}\}_{\text{e}}$), classifying, on average, three more documents in each fold of the cross validation when compared to the best existing solution in the literature.

When using a different letter rather than "e", such as the letter "a", we also see an improvement in the results. The use of multiple representations of letter "a" ($\{\mathcal{S}^{\text{raw}}$,$\mathcal{S}^{\text{med}}$, $\mathcal{S}^{\text{avg}}\}_{\text{a}}$) enables to classify a mean of four more documents in each fold when compared to state-of-the-art techniques. The multiple representation of multiple data "a" and "e" ($\{\mathcal{S}^{\text{raw}}, \mathcal{S}^{\text{med}}, \mathcal{S}^{\text{avg}}\}_{\text{a,e}}$) shows its efficacy by showing the best overall accuracy of the experiments (97.33%), classifying six more documents than the best existing counterpart in the literature, on average. The reason for this good performance relies on the fact that this method takes into account multiple data with different banding artifacts that can be better highlighted using different representations in the specialized deep networks.

To validate the efficacy of the proposed methods, we also performed statistical tests. The Friedmann test showed a p-value of 3.16×10^{-138}, which helps us to state that the difference amongst the methods' performance is statistically significant.Table 4.14 shows the Tukey-HSD pairwise tests.

Considering the best performing configuration of our algorithm ($\{\mathcal{S}^{\text{raw}}, \mathcal{S}^{\text{med}}, \mathcal{S}^{\text{avg}}\}_{\text{a,e}}$)

Table 4.14: Tukey-HSD pairwise statistical test results comparing the proposed methods to the existing ones in the literature.

Rank	Method	$\{S^{raw}, S^{med}, S^{avg}\}_{a,e}$	$\{S^{raw}, S^{med}, S^{avg}\}_{a}$	$\{S^{raw}, S^{med}, S^{avg}\}_{a,e,d}$	$\{S^{raw}, S^{med}, S^{avg}\}_{e}$	CTGF-GLCM-MD-MS [12]	$S^{4\text{-}Conv,raw}_{e}$	$S^{3\text{-}Conv,raw}_{e}$	GLCM-MD-MS [12]	GLCM-MD [12]	HOG [50]	LBP [49]	RECONST-ERR [11]	GLCM [10,32]	CTGF [12]	TOTAL
1	$\{S^{raw}, S^{med}, S^{avg}\}_{a,e}$	0	0	0	0	1	1	1	1	1	1	1	1	1	1	10
2	$\{S^{raw}, S^{med}, S^{avg}\}_{a}$	0	0	0	0	1	1	0	1	1	1	1	1	1	1	9
3	$\{S^{raw}, S^{med}, S^{avg}\}_{a,e,d}$	0	0	0	0	1	1	0	1	1	1	1	1	1	1	9
4	$\{S^{raw}, S^{med}, S^{avg}\}_{e}$	0	0	0	0	1	1	0	1	1	1	1	1	1	1	9
5	CTGF-GLCM-MD-MS [12]	-1	-1	-1	-1	0	0	0	0	1	1	1	1	1	1	2
6	$S^{4\text{-}Conv,raw}_{e}$	-1	-1	-1	-1	0	0	0	0	1	1	1	1	1	1	2
7	$S^{3\text{-}Conv,raw}_{e}$	-1	0	0	0	0	0	0	1	1	1	1	1	1	1	6
8	GLCM-MD-MS [12]	-1	-1	-1	-1	0	0	-1	0	0	0	1	1	1	1	-1
9	GLCM-MD [12]	-1	-1	-1	-1	-1	-1	-1	0	0	0	0	1	1	1	-4
10	HOG [50]	-1	-1	-1	-1	-1	-1	-1	0	0	0	0	1	1	0	-5
11	LBP [49]	-1	-1	-1	-1	-1	-1	-1	-1	0	0	0	1	0	0	-7
12	RECONST-ERR [11]	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	0	-11
13	GLCM [10,32]	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	0	0	0
14	CTGF [12]	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	0	0	0	0

1 = Line method is better than column method
0 = Line method is equivalent to column method
-1 = Line method is worse than column method

and the best literature approach (`CTGF-GLCM-MD-MS`), Table 4.15 and Table 4.16 show confusion matrices representing the classification accuracies per printer. In Table 4.15, the confusion matrix of the proposed method shows that it is possible to identify 100% of three out of ten printers used in the experiments. These printers are Canon MF4370DN, OKI Data C330DN and Samsung CLP315. The `CTGF-GLCM-MD-MS` confusion matrix in Table 4.16, on the other hand, shows 100% classification for only one printer, the OKI Data C330DN.

It is also remarkable the fact that we are using two printers of the same model and brand (H225A and H225B) and it is possible to see, in Table 4.15 and Table 4.16, that there are just some misclassifications between them. The errors in these cases are likely related to the printing artifacts generated by these two printers, which are similar for some documents. The proposed approach misclassified an average of 6.6% of the documents in these two classes, while the best existing method in the literature did it for 7.7% of the documents. It is also important to note that there are some misclassifications when

Table 4.15: Confusion Matrix of the best proposed approach ($\{\mathcal{S}^{raw}, \mathcal{S}^{med}, \mathcal{S}^{avg}\}_{a,e}$) showing, in percentages, the right and wrong mean hits per printer after the 5 × 2 cross validation.

BEST PROPOSED		Attributed Printer									
		B4070	C1150	C3240	C4370	H1518	H225A	H225B	LE260	OC330	SC315
Actual Printer	B4070	99.50		0.33	0.17						
	C1150	0.52	99.48								
	C3240	0.67		98.83	0.50						
	C4370				100.00						
	H1518	0.33	10.50			89.17					
	H225A						93.10	6.90			
	H225B	0.18					6.37	93.45			
	LE260			0.17			0.33		99.50		
	OC330									100.00	
	SC315										100.00

Table 4.16: Confusion Matrix of the best literature solution showing, in percentages, the right and wrong mean hits per printer after the 5 × 2 cross validation.

CTGF-GLCM-MD-MS [12]		Attributed Printer									
		B4070	C1150	C3240	C4370	H1518	H225A	H225B	LE260	OC330	SC315
Actual Printer	B4070	98.67	0.33	1.00							
	C1150	1.72	98.28								
	C3240			97.83	2.17						
	C4370		1.00	0.50	98.50						
	H1518	1.33	10.33			86.83	0.50			0.84	0.17
	H225A		0.50				96.98	2.52			
	H225B						12.90	87.10			
	LE260		0.67	0.50			0.17		98.66		
	OC330									100.00	
	SC315		0.50		0.33						99.17

classifying printers H1518 (an HP printer) and C3240 (a Canon printer) in both cases. This happens because these two printers present a slightly smaller average font size with respect to the other eight, as can be seen in Figure 4.18. Therefore, they probably share some common artifacts.

4.2.5 Conclusions

Laser printer attribution is a difficult task that involves investigating several printing patterns, created with different manufacturing processes, models and brands. Existing methods in the literature rely on computer vision and machine learning algorithms applied to scanned versions of documents, aiming at finding intrinsic signatures on printed material that better discriminate different printers. The main problem with these approaches is that they are underpinned by so-called hand-crafted features, which often require expert domain-knowledge to properly capture discriminative artifacts useful in the attribution process (*e.g.*, intrinsic texture, geometric distortions in the printed material, etc.). Ideally, it would be interesting to also be able to detect important discriminative features directly from training data (data-driven methods). Those features could be even

combined with hand-crafted ones for a more effective method.

In this vein, in this work, we have proposed a solution capable of learning discriminative features for the printer attribution problem directly from available training data (*i.e.,* scanned versions of printed papers). The solution inherits the benefits of convolutional neural networks and back-propagation procedures, evolving the descriptor during training, thus making these networks tailored to the analyzed data. The method relies on artifacts captured from different letters of documents in different languages. It also uses other letters rather than the commonly used "e". To better highlight characteristic artifacts, different data representations through some image transformations were also investigated.

As we discussed thoroughly in this work, the use of multiple representations of multiple data allows to outperform the state of the art when for the laser printer attribution problem. Multiple representations fed as input to the used deep networks are important because they highlight different characteristics of the input images. We also showed that multiple representations of multiple data is a reasonable choice for laser printer attribution with deep networks. Indeed, with the benefits of the multiple representations presented before, multiple data also ensures a larger amount of voters per document.

One interesting finding in this research is the promising use of these different representations, composed by low-pass filtering residuals, as input to Convolutional Neural Networks. In a real-world setup, in which a suspect document was printed using a toner different from the one used for training the method, these low-pass filtering residuals can work better for pointing out the source than raw image inputs, as this last representation is more affected by the change of toner due to the increased presence of high-frequency components linked to toner artifacts. One natural extension of this proposed approach for this cross-dataset setup is replacing the raw image representation with other low-pass filtering residual analyses, such as the Gaussian filtering residual [229], bilateral filtering [230] and guided image filtering [231].

With current solutions to the printer attribution problem achieving high classification results, we believe it is time to aim at more challenging scenarios. For instance, current methods in the literature have shown great potential for classifying documents printed in similar conditions (both physical but also temporally close together. As a matter of fact, the printer attribution problem is much more difficult than its related problem of sensor attribution (for cameras and scanners). The reason is that the printing process has much more mechanical elements involved and intertwined when printing a document. Such elements surely play different roles in the creation of a unique signature for each printer. However, and the literature needs more study in this regard, it is natural that such

signature will not last forever and will probably degrade over time as different elements in the printer age and defects appear. Then the next question is what happens if a document was printed several years ago, while the printer under suspicion was just recently acquired by the investigators. A thorough investigation of this problem considering data captured in several moments along the years will be a significant contribution to the field.

4.3 Conclusions

In this chapter we showed two multimedia forensics applications that benefit from the adoption of data-driven methods. A set of vision tools and deep-learning frameworks, mainly developed by the computer vision community, is studied and fused with hand-crafted features extraction and signal processing techniques to improve state-of-the-art performance in single vs. double JPEG detection and laser printer attribution. For both applications the fusion of several deep-learning pipelines with proper pre-processing techniques improves the results with respect to a blind usage of data-driven methods on raw data.

The lesson learned is that data-driven techniques alone might suffer in multimedia forensics applications when not paired with a proper domain knowledge, necessary to understand the observed phenomena and process the available input data to allow a better learning from a limited amount of training data.

5 Conclusions

In this thesis we faced several open challenges in the multimedia forensics field. Most of the proposed methods exploit a combination of classic signal processing techniques, handcrafted features and data-driven methods. All the methods and results included in this thesis have been published in peer-reviewed conferences and journals in the last three years.

In Chapter 2 we proposed the first data-driven method for camera model identification. A family of convolutional neural networks is designed and tested under several data-splitting policies, evaluating the effect of shallow rather then deep architectures. We designed and trained a CNN architecture that extracts meaningful forensics features from small image patches, thus paving the way to a fine-grained analyses of digital images. The challenging choice of working on small image patches is motivated by the perspective of detecting intra-image inconsistencies. In facts, the features extracted with the proposed CNN architecture are exploited in an iterative clustering scheme to face the problems of tampering detection and localization. For each image patch we define a quality function to feed the clustering algorithm with a confidence measure of the provided camera-model-based feature. We concluded the chapter with two CNN-tailored attacks that are proposed and tested as antiforensics methods to fool a deep-learning based camera identification system. The effectiveness of such attacks poses new challenges in a real-world deployment of data-driven systems for source attribution. Despite the promising results of data-driven methods for camera model identification, the lack of extended, shared, and challenging evaluation databases and metrics limits the portability of such systems to the industry. Challenges as working in an open-set scenario, where not all the camera models are known, or under multiple global images modifications still require a lot of effort to be solved.

In Chapter 3 we studied the problem of PRNU fingerprints and residuals compression, a step toward large-scale adoption of sensor-based source identification. We faced the topic by defining a design methodology and a compression pipeline tailored at reducing the intrinsic redundancies of PRNU while preserving a high detection rate. The two presented approaches face the problem of PRNU compression under two different

perspectives: computational complexity and bitrate reduction. The designed projection matrices provide an impressive reduction in terms of computational complexity while preserving state-of-the-art detection performance. The proposed pipeline based on decimation, Random Projections and dead-zone quantization provides state-of-the-art performance in terms of bitrate vs. detection rate. We also proposed two types of antiforensics attacks tailored to PRNU removal. A first no-reference approach leverages inpainting techniques to reconstruct each pixel starting from its neighbors. A second reference-based data-driven approach is designed as a double-input autoencoder properly trained on a single image to attenuate PRNU traces while preserving an excellent image quality. PRNU fingerprints and residuals extracted from smartphones cameras and video frames are finally exploited in two forensics applications, to create a sensor-based fingerprint for smartphone authentication and to pre-process collections of videos for multimedia phylogeny applications. Several PRNU-related challenges are still open, as the correct extraction of fingerprints and residuals from stabilized video sequences and high dynamic range images.

We concluded in Chapter 4 with two forensics applications, where a combination of handcrafted features and data-learned features is the key to solve the task at hand. Single versus double JPEG detection is faced with a combination of handcrafted features extraction and properly designed CNN architectures. Similarly, laser printer attribution based on data-driven method benefits from classic signal pre-processing operations on the raw input data.

The increasing interest from both researches and companies to multimedia forensics applications and solutions is surely a positive element toward the development of advanced, reliable and scalable real-world systems. The lack of a common evaluation protocol for the many forensics tasks faced by the scientific community is still an open issue. We believe that in the next few years the resources devoted to this field will allow the collection of large and challenging publicly available datasets to measure, as objectively as possible, the performance of the many proposed forensics systems.

The main lesson learned from the set of works carried out over the past few years is that a blind transfer of machine learning and data-driven techniques to the multimedia forensics field is not enough to perform at par with handcrafted state-of-the-art solutions. The knowledge of the domain and of the signals at hand is key to turn a powerful data-learning tool into an effective forensic tool. The fusion of the "best of the two worlds" seems to be the way to go when dealing with limited training data and challenging forensics tasks.

Bibliography

[1] Anderson Rocha, Walter Scheirer, Terrance Boult, and Siome Goldenstein, "Vision of the unseen," *ACM Comput. Surv.*, vol. 43, no. 4, pp. 1–42, oct 2011.

[2] H. T. Sencar and N. Memon, "Overview of State-of-the-Art in Digital Image Forensics," in *Algorithms, Archit. Inf. Syst. Secur.*, pp. 325–347. nov 2008.

[3] Hany Farid, "Exposing digital forgeries in scientific images," in *Proceeding 8th Work. Multimed. Secur. - MM&Sec '06*, New York, New York, USA, 2006, p. 29, ACM Press.

[4] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, "Generative Adversarial Nets," *Proc. Neural Inf. Process. Syst. Conf.*, pp. 2672–2680, dec 2014.

[5] Iryna Korshunova, Wenzhe Shi, Joni Dambre, and Lucas Theis, "Fast Face-Swap Using Convolutional Neural Networks," in *2017 IEEE Int. Conf. Comput. Vis.* oct 2017, pp. 3697–3705, IEEE.

[6] Matthew C. Stamm, Min Wu, and K. J. Ray Liu, "Information Forensics: An Overview of the First Decade," *IEEE Access*, vol. 1, pp. 167–200, 2013.

[7] Alessandro Piva, "An Overview on Image Forensics," *ISRN Signal Process.*, vol. 2013, pp. 1–22, 2013.

[8] P. Bestagini, K. M. Fontani, S. Milani, M. Barni, A. Piva, M. Tagliasacchi, and K. S. Tubaro, "An overview on video forensics," *Eur. Signal Process. Conf.*, vol. 1, no. 2012, pp. 1229–1233, 2012.

[9] Christian Böhm, Stefan Berchtold, and Daniel A Keim, "Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases," *ACM Comput. Surv.*, vol. 33, no. 3, pp. 322–373, sep 2001.

[10] Luca Bondi, Luca Baroffio, David Guera, Paolo Bestagini, Edward J. Delp, and Stefano Tubaro, "First Steps Toward Camera Model Identification With Convolu-

tional Neural Networks," *IEEE Signal Process. Lett.*, vol. 24, no. 3, pp. 259–263, mar 2017.

[11] Luca Bondi, Silvia Lameri, David Guera, Paolo Bestagini, Edward J. Delp, and Stefano Tubaro, "Tampering Detection and Localization Through Clustering of Camera-Based CNN Features," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. jul 2017, pp. 1855–1864, IEEE.

[12] David Guera, Yu Wang, Luca Bondi, Paolo Bestagini, Stefano Tubaro, and Edward J Delp, "A Counter-Forensic Method for CNN-Based Camera Model Identification," *2017 IEEE Conf. Comput. Vis. Pattern Recognit. Work.*, vol. 2017-July, pp. 1840–1847, jul 2017.

[13] Jan Lukas, Jessica Fridrich, and Miroslav Goljan, "Determining digital image origin using sensor imperfections," in *Proceedings of the SPIE*, Amir Said and John G. Apostolopoulos, Eds., mar 2005, vol. 5685, p. 249.

[14] Luca Bondi, David Güera, Luca Baroffio, Paolo Bestagini, EdwardJ Delp, and Stefano Tubaro, "A Preliminary Study on Convolutional Neural Networks for Camera Model Identification," in *Electron. Imaging*, jan 2017, vol. 2017, pp. 67–76.

[15] Diego Valsesia, Giulio Coluccia, Tiziano Bianchi, and Enrico Magli, "Compressed Fingerprint Matching and Camera Identification via Random Projections," *IEEE Trans. Inf. Forensics Secur.*, vol. 10, no. 7, pp. 1472–1485, jul 2015.

[16] Luca Bondi, Paolo Bestagini, Fernando Perez-Gonzalez, and Stefano Tubaro, "Improving PRNU Compression through Preprocessing, Quantization and Coding," *IEEE Trans. Inf. Forensics Secur.*, pp. 1–1, 2018.

[17] Luca Bondi, Fernando Pérez-González, Paolo Bestagini, and Stefano Tubaro, "Design of projection matrices for PRNU compression," in *2017 IEEE Workshop on Information Forensics and Security (WIFS)*. dec 2017, pp. 1–6, IEEE.

[18] Irene Amerini, Paolo Bestagini, Luca Bondi, Roberto Caldelli, Matteo Casini, and Stefano Tubaro, "Robust smartphone fingerprint by mixing device sensors features for mobile strong authentication," *Electron. Imaging*, vol. 2016, no. 8, pp. 1–8, feb 2016.

[19] Silvia Lameri, Luca Bondi, Paolo Bestagin, and Stefano Tubaro, "Near-duplicate video detection exploiting noise residual traces," in *2017 IEEE Int. Conf. Image Process.* sep 2017, pp. 1497–1501, IEEE.

[20] John Entrieri and Matthias Kirchner, "Patch-Based Desynchronization of Digital Camera Sensor Fingerprints," *Electronic Imaging*, vol. 2016, no. 8, pp. 1–9, feb 2016.

[21] Sara Mandelli, Luca Bondi, Silvia Lameri, Vincenzo Lipari, Paolo Bestagini, and Stefano Tubaro, "Inpainting-Based camera anonymization," in *2017 IEEE Int. Conf. Image Process.* sep 2017, vol. 2017-Septe, pp. 1522–1526, IEEE.

[22] Nicolò Bonettini, Sara Mandelli, Luca Bondi, Paolo Bestagini, Stefano Tubaro, David Düera, and Edward J. Delp, "Fooling PRNU-Based Detectors Through Convolutional Neural Networks," in *2018 26th Eur. Signal Process. Conf.*, 2018.

[23] Mauro Barni, Luca Bondi, Nicolò Bonettini, Paolo Bestagini, Andrea Costanzo, Marco Maggini, Benedetta Tondi, and Stefano Tubaro, "Aligned and non-aligned double JPEG detection using convolutional neural networks," *J. Vis. Commun. Image Represent.*, vol. 49, pp. 153–163, nov 2017.

[24] Anselmo Ferreira, Luca Bondi, Luca Baroffio, Paolo Bestagini, Jiwu Huang, Jefersson A. dos Santos, Stefano Tubaro, and Anderson Rocha, "Data-Driven Feature Characterization Techniques for Laser Printer Attribution," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 8, pp. 1860–1873, aug 2017.

[25] Kaitai Liang, Joseph K Liu, Rongxing Lu, and Duncan S Wong, "Privacy Concerns for Photo Sharing in Online Social Networks," *IEEE Internet Computing*, vol. 19, no. 2, pp. 58–63, mar 2015.

[26] Hany Farid, "Seeing is not believing," *IEEE Spectrum*, vol. 46, no. 8, pp. 44–51, aug 2009.

[27] Mehdi Kharrazi, H.T. Sencar, and Nasir Memon, "Blind source camera identification," in *2004 International Conference on Image Processing, 2004. ICIP '04.* oct 2004, vol. 1, pp. 709–712, IEEE.

[28] Tomas Filler, Jessica Fridrich, and Miroslav Goljan, "Using sensor pattern noise for camera model identification," in *2008 15th IEEE International Conference on Image Processing.* 2008, number 1, pp. 1296–1299, IEEE.

[29] Matthias Kirchner and Thomas Gloe, "Forensic Camera Model Identification," in *Handbook of Digital Forensics of Multimedia Data and Devices*, pp. 329–374. John Wiley & Sons, Ltd, Chichester, UK, dec 2015.

[30] Ashwin Swaminathan, Min Wu, and K.J. Ray Liu, "Digital image forensics via intrinsic fingerprints," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 1, pp. 101–117, mar 2008.

[31] Davide Cozzolino, Diego Gragnaniello, and Luisa Verdoliva, "Image forgery localization through the fusion of camera-based, feature-based and pixel-based techniques," in *2014 IEEE International Conference on Image Processing (ICIP)*. oct 2014, pp. 5302–5306, IEEE.

[32] Lorenzo Gaborini, Paolo Bestagini, Simone Milani, Marco Tagliasacchi, and Stefano Tubaro, "Multi-Clue Image Tampering Localization," in *2014 IEEE International Workshop on Information Forensics and Security (WIFS)*. dec 2014, pp. 125–130, IEEE.

[33] Yann Le Cun and Yoshua Bengio, *The Handbook of Brain Theory and Neural Networks*, MIT Press, 1998.

[34] Y Bengio, "Learning Deep Architectures for AI," *Foundations and Trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[35] Alex Krizhevsky, IIya Sulskever, and Geoffret E Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information and Processing Systems (NIPS)*, pp. 1–9, 2012.

[36] Min Lin, Qiang Chen, and Shuicheng Yan, "Network In Network," *arXiv preprint*, p. 10, dec 2013.

[37] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. jun 2015, pp. 1–9, IEEE.

[38] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf, "DeepFace: Closing the Gap to Human-Level Performance in Face Verification," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*. jun 2014, pp. 1701–1708, IEEE.

[39] Jan Hosang, Mohamed Omran, Rodrigo Benenson, and Bernt Schiele, "Taking a deeper look at pedestrians," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. jun 2015, pp. 4073–4082, IEEE.

[40] Denis Tomè, Federico Monti, Luca Baroffio, Luca Bondi, Marco Tagliasacchi, and Stefano Tubaro, "Deep Convolutional Neural Networks for pedestrian detection," *Signal Process. Image Commun.*, vol. 47, pp. 482–489, sep 2016.

[41] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei, "Large-Scale Video Classification with Convolutional Neural Networks," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1725–1732, jun 2014.

[42] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, may 2015.

[43] Daniel Mas Montserrat, Qian Lin, Jan Allebach, and EdwardJ. Delp, "Training Object Detection And Recognition CNN Models Using Data Augmentation," *Electronic Imaging*, vol. 2017, no. 10, pp. 27–36, jan 2017.

[44] Da Luo, Rui Yang, and Jiwu Huang, "Detecting double compressed AMR audio using deep learning," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. may 2014, pp. 2669–2673, IEEE.

[45] Jiansheng Chen, Xiangui Kang, Ye Liu, and Z Jane Wang, "Median Filtering Forensics Based on Convolutional Neural Networks," *IEEE Signal Processing Letters*, vol. 22, no. 11, pp. 1849–1853, nov 2015.

[46] Yinlong Qian, Jing Dong, Wei Wang, and Tieniu Tan, "Deep learning for steganalysis via convolutional neural networks," in *Proceedings of SPIE - The International Society for Optical Engineering*, Adnan M. Alattar, Nasir D. Memon, and Chad D. Heitzenrater, Eds. mar 2015, vol. 9409, p. 94090J, International Society for Optics and Photonics.

[47] Belhassen Bayar and Matthew C. Stamm, "A Deep Learning Approach to Universal Image Manipulation Detection Using a New Convolutional Layer," in *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security - IH&MMSec '16*, New York, New York, USA, 2016, pp. 5–10, ACM Press.

[48] Guanshuo Xu, Han-Zhou Wu, and Yun-Qing Shi, "Structural Design of Convolutional Neural Networks for Steganalysis," *IEEE Signal Processing Letters*, vol. 23, no. 5, pp. 708–712, may 2016.

[49] Matthias Kirchner and Rainer Böhme, "Tamper Hiding: Defeating Image Forensics," *Proceedings of the International Workshop on Information Hiding*, pp. 326–341, jun 2007.

[50] Miroslav Goljan, Jessica Fridrich, and Tomáš Filler, "Managing a large database of camera fingerprints," in *Is&T/Spie*, Nasir D. Memon, Jana Dittmann, Adnan M. Alattar, and Edward J. Delp III, Eds., feb 2010, vol. 7541, p. 754108.

[51] Thomas Gloe, Matthias Kirchner, Antje Winkler, and Rainer Böhme, "Can we trust digital image forensics?," in *Proceedings of the 15th international conference on Multimedia*, New York, New York, USA, 2007, MM '07, p. 78, ACM Press.

[52] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus, "Intriguing properties of neural networks," *Proceedings of the International Conference on Learning Representations*, apr 2014.

[53] I J Goodfellow, J Shlens, and C Szegedy, "Explaining And Harnessing Adversarial Examples," *Proceedings of the International Conference on Learning Representations*, may 2015.

[54] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami, "Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks," *2016 IEEE Symposium on Security and Privacy (SP)*, pp. 582–597, may 2016.

[55] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami, "The Limitations of Deep Learning in Adversarial Settings," *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 372–387, mar 2016.

[56] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger, "Densely Connected Convolutional Networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, jul 2017.

[57] Kai San Choi, Edmund Y Lam, and Kenneth K Y Wong, "Automatic source camera identification using the intrinsic lens radial distortion," *Optics Express*, vol. 14, no. 24, pp. 11551, 2006.

[58] Sevinc Bayram, H. Sencar, Nasir Memon, and Ismail Avcibas, "Source camera identification based on CFA interpolation," in *IEEE International Conference on Image Processing 2005*. 2005, vol. 3, pp. III–69, IEEE.

[59] Shruti Agarwal and Hany Farid, "Photo forensics from JPEG dimples," in *2017 IEEE Workshop on Information Forensics and Security (WIFS)*. dec 2017, pp. 1–6, IEEE.

[60] Léon Bottou, "Large-Scale Machine Learning with Stochastic Gradient Descent," *Proceedings of COMPSTAT'2010*, pp. 177–186, aug 2010.

[61] Yoshua Bengio, "Practical recommendations for gradient-based training of deep architectures," jun 2012.

[62] Erwin Quiring and Matthias Kirchner, "Fragile sensor fingerprint camera identification," in *2015 IEEE International Workshop on Information Forensics and Security (WIFS)*. nov 2015, pp. 1–6, IEEE.

[63] Lionel Pibre, Pasquet Jérôme, Dino Ienco, and Marc Chaumont, "Deep Learning for steganalysis is better than a Rich Model with an Ensemble Classifier, and is natively robust to the cover source-mismatch," in *arXiv preprint arXiv:1511.04855*, 2016.

[64] Vahid Sedighi and Jessica Fridrich, "Histogram Layer, Moving Convolutional Neural Networks Towards Feature-Based Steganalysis," in *Electronic Imaging*, jan 2017, vol. 2017, pp. 50–55.

[65] I. Avcibas, N. Memon, and B. Sankur, "Steganalysis using image quality metrics," *IEEE Transactions on Image Processing*, vol. 12, no. 2, pp. 221–229, feb 2003.

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

[67] Hong Cao and A.C. Kot, "Accurate Detection of Demosaicing Regularity for Digital Image Forensics," *IEEE Transactions on Information Forensics and Security*, vol. 4, no. 4, pp. 899–910, dec 2009.

[68] Guanshuo Xu and Yun Qing Shi, "Camera Model Identification Using Local Binary Patterns," in *2012 IEEE International Conference on Multimedia and Expo*. jul 2012, pp. 392–397, IEEE.

[69] Simone Milani, Paolo Bestagini, Marco Tagliasacchi, and Stefano Tubaro, "Demosaicing strategy identification via eigenalgorithms," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. may 2014, pp. 2659–2663, IEEE.

[70] Thanh Hai Thai, Remi Cogranne, and Florent Retraint, "Camera Model Identification Based on the Heteroscedastic Noise Model," *IEEE Transactions on Image Processing*, vol. 23, no. 1, pp. 250–263, jan 2014.

[71] Jessica Fridrich and Jan Kodovsky, "Rich Models for Steganalysis of Digital Images," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 868–882, jun 2012.

[72] Chen Chen and Matthew C. Stamm, "Camera model identification framework using an ensemble of demosaicing features," in *2015 IEEE International Workshop on Information Forensics and Security (WIFS)*. nov 2015, pp. 1–6, IEEE.

[73] Francesco Marra, Giovanni Poggi, Carlo Sansone, and Luisa Verdoliva, *Evaluation of Residual-Based Local Features for Camera Model Identification*, pp. 11–18, Springer International Publishing, 2015.

[74] Amel Tuama, Frederic Comby, and Marc Chaumont, "Camera model identification based machine learning approach with high order statistics features," in *2016 24th European Signal Processing Conference (EUSIPCO)*. aug 2016, pp. 1183–1187, IEEE.

[75] Markos Zampoglou, Symeon Papadopoulos, and Yiannis Kompatsiaris, "Large-scale evaluation of splicing localization algorithms for web images," *Multimedia Tools and Applications*, vol. 76, no. 4, pp. 4801–4834, feb 2017.

[76] Zhouchen Lin, Junfeng He, Xiaoou Tang, and Chi-Keung Tang, "Fast, automatic and fine-grained tampered JPEG image detection via DCT coefficient analysis," *Pattern Recognition*, vol. 42, no. 11, pp. 2492–2501, nov 2009.

[77] Weihai Li, Yuan Yuan, and Nenghai Yu, "Passive detection of doctored JPEG image via block artifact grid extraction," *Signal Processing*, vol. 89, no. 9, pp. 1821–1829, sep 2009.

[78] Pasquale Ferrara, Tiziano Bianchi, Alessia De Rosa, and Alessandro Piva, "Image Forgery Localization via Fine-Grained Analysis of CFA Artifacts," *IEEE Trans-*

actions on Information Forensics and Security, vol. 7, no. 5, pp. 1566–1577, oct 2012.

[79] A.E. Dirik, H.T. Sencar, and N Memon, "Digital Single Lens Reflex Camera Identification From Traces of Sensor Dust," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 3, pp. 539–552, sep 2008.

[80] Shuiming Ye, Qibin Sun, and Ee-Chien Chang, "Detecting Digital Image Forgeries by Measuring Inconsistencies of Blocking Artifact," in *Multimedia and Expo, 2007 IEEE International Conference on*. jul 2007, pp. 12–15, IEEE.

[81] Neal Krawetz, "A Picture's Worth...," *Hacker Factor Solutions*, pp. 1–31, 2007.

[82] Hany Farid, "Exposing Digital Forgeries From JPEG Ghosts," *IEEE Transactions on Information Forensics and Security*, vol. 4, no. 1, pp. 154–160, mar 2009.

[83] Babak Mahdian and Stanislav Saic, "Using noise inconsistencies for blind image forensics," *Image and Vision Computing*, vol. 27, no. 10, pp. 1497–1503, sep 2009.

[84] Marco Fontani, Tiziano Bianchi, Alessia De Rosa, Alessandro Piva, and Mauro Barni, "A Framework for Decision Fusion in Image Forensics Based on Dempster-Shafer Theory of Evidence," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 4, pp. 593–607, apr 2013.

[85] Davide Cozzolino, Giovanni Poggi, and Luisa Verdoliva, "Splicebuster: A new blind image splicing detector," in *2015 IEEE International Workshop on Information Forensics and Security (WIFS)*. nov 2015, pp. 1–6, IEEE.

[86] Davide Cozzolino and Luisa Verdoliva, "Single-image splicing localization through autoencoder-based anomaly detection," in *2016 IEEE International Workshop on Information Forensics and Security (WIFS)*. dec 2016, pp. 1–6, IEEE.

[87] Dario D'Avino, Davide Cozzolino, Giovanni Poggi, and Luisa Verdoliva, "Autoencoder with recurrent neural networks for video forgery detection," *Electronic Imaging*, vol. 2017, no. 7, pp. 92–99, jan 2017.

[88] Wei-Hong Chuang and Min Wu, "Robustness of Color Interpolation Identification against Anti-forensic Operations," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7692 LNCS, pp. 16–30. 2013.

[89] Matthias Kirchner and Rainer Böhme, "Synthesis of color filter array pattern in digital images," feb 2009, p. 72540K.

[90] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna, "Rethinking the Inception Architecture for Computer Vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. jun 2016, pp. 2818–2826, IEEE.

[91] Karen Simonyan and Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *Proceedings of the International Conference on Learning Representations*, 2015.

[92] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. jun 2016, pp. 770–778, IEEE.

[93] O Russakovsky, J Deng, H Su, J Krause, S Satheesh, S Ma, Z Huang, A Karpathy, A Khosla, M Bernstein, A C Berg, and L Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, dec 2015.

[94] Thomas Gloe and Rainer Böhme, "The Dresden Image Database for Benchmarking Digital Image Forensics," *Journal of Digital Forensic Practice*, vol. 3, no. 2-4, pp. 150–159, dec 2010.

[95] Nicolas Papernot, Ian Goodfellow, Ryan Sheatsley, Reuben Feinman, and Patrick McDaniel, "cleverhans v1.0.0: an adversarial machine learning library," *arXiv:1610.00768*, oct 2016.

[96] Xiaoyu Feng, Hongting Zhang, Hsiao-Chun Wu, and Yiyan Wu, "A New Approach for Optimal Multiple Watermarks Injection," *IEEE Signal Processing Letters*, vol. 18, no. 10, pp. 575–578, oct 2011.

[97] Julien Voisin, Christophe Guyeux, and Jacques M Bahi, "The Metadata Anonymization Toolkit," 2017.

[98] Lanh Tran Van, Sabu Emmanuel, and Mohan S Kankanhalli, "Identifying Source Cell Phone using Chromatic Aberration," in *Multimedia and Expo, 2007 IEEE International Conference on*. jul 2007, pp. 883–886, IEEE.

[99] Ashwin Swaminathan, Min Wu, and K.J.R. Liu, "Nonintrusive component forensics of visual sensors using output images," *IEEE Transactions on Information Forensics and Security*, vol. 2, no. 1, pp. 91–106, mar 2007.

[100] Erwin J. Alles, Zeno J M H Geradts, and Cor J. Veenman, "Source Camera Identification for Heavily JPEG Compressed Low Resolution Still Images," *Journal of Forensic Sciences*, vol. 54, no. 3, pp. 628–638, may 2009.

[101] O Celiktutan, B Sankur, and I Avcibas, "Blind Identification of Source Cell-Phone Model," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 3, pp. 553–566, sep 2008.

[102] Miroslav Goljan, Mo Chen, and Jessica Fridrich, "Identifying Common Source Digital Camera from Image Pairs," in *2007 IEEE International Conference on Image Processing*. 2007, vol. 6, pp. VI – 125–VI – 128, IEEE.

[103] Mo Chen, Jessica Fridrich, Miroslav Goljan, and Jan Lukas, "Determining Image Origin and Integrity Using Sensor Noise," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 1, pp. 74–90, 2008.

[104] J. Fridrich, "Digital image forensics," *IEEE Signal Processing Magazine*, vol. 26, no. 2, pp. 26–37, mar 2009.

[105] J. Luka, Jessica Fridrich, and Miroslav Goljan, "Digital Camera Identification From Sensor Pattern Noise," *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 2, pp. 205–214, jun 2006.

[106] Irene Amerini, Roberto Caldelli, Vito Cappellini, Francesco Picchioni, and Alessandro Piva, "Analysis of denoising filters for photo response non uniformity noise extraction in source camera identification," in *2009 16th International Conference on Digital Signal Processing*. jul 2009, pp. 1–7, IEEE.

[107] Giovanni Chierchia, Sara Parrilli, Giovanni Poggi, Carlo Sansone, and Luisa 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 - MiFor '10*, New York, New York, USA, 2010, p. 117, ACM Press.

[108] A Cortiana, V Conotter, G Boato, and F. G. B. De Natale, "Performance comparison of denoising filters for source camera identification," in *Proceedings of the SPIE Conference on Media Watermarking, Security, and Forensics*, Nasir D.

Memon, Jana Dittmann, Adnan M. Alattar, and Edward J. Delp III, Eds., feb 2011, p. 788007.

[109] Floris Gisolf, Anwar Malgoezar, Teun Baar, and Zeno Geradts, "Improving source camera identification using a simplified total variation based noise removal algorithm," *Digital Investigation*, vol. 10, no. 3, pp. 207–214, oct 2013.

[110] Xufeng Lin and Chang-Tsun Li, "Preprocessing Reference Sensor Pattern Noise via Spectrum Equalization," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 1, pp. 126–140, jan 2016.

[111] Chang-Tsun Li, "Source Camera Identification Using Enhanced Sensor Pattern Noise," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 2, pp. 280–287, jun 2010.

[112] Miroslav Goljan and Jessica Fridrich, "Camera identification from cropped and scaled images," in *SPIE Electronic Imaging, Forensics, Security, Steganography, and Watermarking of Multimedia Contents*, Edward J. Delp III, Ping Wah Wong, Jana Dittmann, and Nasir D. Memon, Eds., feb 2008, p. 68190E.

[113] Kurt Rosenfeld and Husrev T. Sencar, "A study of the robustness of PRNU-based camera identification," in *IS&T/SPIE Electronic Imaging (EI)*, Edward J. Delp III, Jana Dittmann, Nasir D. Memon, and Ping Wah Wong, Eds. International Society for Optics and Photonics, feb 2009, vol. 7254, p. 72540M.

[114] Miroslav Goljan, Jessica Fridrich, and Tomáš Filler, "Large scale test of sensor fingerprint camera identification," in *SPIE Media Forensics and Security*, Edward J. Delp III, Jana Dittmann, Nasir D. Memon, and Ping Wah Wong, Eds., feb 2009, p. 72540I.

[115] Roberto Caldelli, Irene Amerini, Francesco Picchioni, and Matteo Innocenti, "Fast image clustering of unknown source images," in *2010 IEEE International Workshop on Information Forensics and Security*. dec 2010, pp. 1–5, IEEE.

[116] Francesco Marra, Giovanni Poggi, Carlo Sansone, and Luisa Verdoliva, "Blind PRNU-Based Image Clustering for Source Identification," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 9, pp. 2197–2211, sep 2017.

[117] Diego Valsesia, Giulio Coluccia, Tiziano Bianchi, and Enrico Magli, "User Authentication via PRNU-Based Physical Unclonable Functions," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 8, pp. 1941–1956, aug 2017.

[118] Miroslav Goljan, Mo Chen, Pedro Comesaña, and Jessica Fridrich, "Effect of Compression on Sensor-Fingerprint Based Camera Identification," in *Electronic Imaging*, feb 2016, vol. 2016, pp. 1–10.

[119] Sevinç Bayram, Hüsrev Taha Sencar, and Nasir Memon, "Efficient Sensor Fingerprint Matching Through Fingerprint Binarization," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 4, pp. 1404–1413, aug 2012.

[120] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang, "Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142–3155, jul 2017.

[121] M. Kivanc Mihcak, I Kozintsev, K Ramchandran, and P Moulin, "Low-complexity image denoising based on statistical modeling of wavelet coefficients," *IEEE Signal Processing Letters*, vol. 6, no. 12, pp. 300–303, dec 1999.

[122] Ahmet Karaküçük and Ahmet Emir Dirik, "Adaptive photo-response non-uniformity noise removal against image source attribution," *Digital Investigation*, vol. 12, pp. 66–76, mar 2015.

[123] Chang-Tsun Li, Chih-Yuan Chang, and Yue Li, "On the Repudiability of Device Identification and Image Integrity Verification Using Sensor Pattern Noise," in *Information Security and Digital Forensics*, Dasun Weerasinghe, Ed., Berlin, Heidelberg, 2010, pp. 19–25, Springer Berlin Heidelberg.

[124] Ahmet Emir Dirik, Husrev Taha Sencar, and Nasir Memon, "Analysis of Seam-Carving-Based Anonymization of Images Against PRNU Noise Pattern-Based Source Attribution," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 12, pp. 2277–2290, dec 2014.

[125] Hui Zeng, Jiansheng Chen, Xiangui Kang, and Wenjun Zeng, "Removing camera fingerprint to disguise photograph source," in *2015 IEEE International Conference on Image Processing (ICIP)*. sep 2015, pp. 1687–1691, IEEE.

[126] Sevinç Bayram, Hüsrev Taha Sencar, and Nasir Memon, "Seam-carving based anonymization against image & video source attribution," in *IEEE International Workshop on Multimedia Signal Processing (MMSP)*, 2013.

[127] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman, "PatchMatch," *ACM SIGGRAPH 2009 papers on - SIGGRAPH '09*, vol. 28, no. 3, pp. 1, aug 2009.

[128] Luca Bondi, Fernando Pérez-González, Paolo Bestagini, and Stefano Tubaro, "Design of Projection Matrices for PRNU Compression - Addendum," Tech. Rep., 2017.

[129] R. Keys, "Cubic convolution interpolation for digital image processing," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, no. 6, pp. 1153–1160, dec 1981.

[130] Konstantinos Papafitsoros, Carola Bibiane Schoenlieb, and Bati Sengul, "Combined first and second order total variation inpainting using split Bregman," *Image Processing On Line (IPOL)*, vol. 3, pp. 112–136, 2013.

[131] Ralph A. Willoughby, *Solutions of Ill-Posed Problems (A. N. Tikhonov and V. Y. Arsenin)*, vol. 21 of *Scripta series in mathematics*, Winston, apr 1979.

[132] Leonid I. Rudin, Stanley Osher, and Emad Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1-4, pp. 259–268, nov 1992.

[133] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian, "Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering," *IEEE Transactions on Image Processing*, vol. 16, no. 8, pp. 2080–2095, aug 2007.

[134] A Paszke, S Gross, S Chintala, G Chanan, E Yang, Z DeVito, Z Lin, A Desmaison, L Antiga, and A Lerer, "Automatic differentiation in {PyTorch}," *NIPS, Autodiff Workshop*, 2017.

[135] Adolfo S. Coronado, *Computer Security: Principles and Practice, Second Edition*, vol. 9, Prentice Hall Press, apr 2013.

[136] Sanorita Dey, Nirupam Roy, Wenyuan Xu, Romit Roy Choudhury, and Srihari Nelakuditi, "AccelPrint: Imperfections of Accelerometers Make Smartphones Trackable," in *Proceedings 2014 Network and Distributed System Security Symposium*, Reston, VA, 2014, Internet Society.

[137] A Das, N Borisov, and M Caesar, "Exploring Ways To Mitigate Sensor-Based Smartphone Fingerprinting," .

[138] Bruno Bertini, Roberto Caldelli, Irene Amerini, and Rudy Becarelli, "Acquisition source identification through a blind image classification," in *IET Image Processing*, apr 2015, vol. 9, pp. 329–337.

[139] Miroslav Goljan and Jessica Fridrich, "Sensor fingerprint digests for fast camera identification from geometrically distorted images," in *SPIE Conference on Media Watermarking, Security, and Forensics*, Adnan M. Alattar, Nasir D. Memon, and Chad D. Heitzenrater, Eds., mar 2013, p. 86650B.

[140] O Lartillot and P Toiviainen, "MIR in Matlab: A toolbox for musical feature extraction from audio," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2007.

[141] P Bestagini, M Zanoni, L Albonico, A Paganini, A Sarti, and S Tubaro, "Feature-based classification for audio bootlegs detection," in *2013 IEEE International Workshop on Information Forensics and Security (WIFS)*. nov 2013, pp. 126–131, IEEE.

[142] Irene Amerini, Roberto Caldelli, Vito Cappellini, Francesco Picchioni, and Alessandro Piva, "Estimate of PRNU Noise Based on Different Noise Models for Source Camera Identification," *International Journal of Digital Crime and Forensics*, vol. 2, no. 2, pp. 21–33, apr 2010.

[143] Tin Kam Ho, "The random subspace method for constructing decision forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, 1998.

[144] Matthew C. Stamm, W. Sabrina Lin, and K. J. Ray Liu, "Temporal Forensics and Anti-Forensics for Motion Compensated Video," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 4, pp. 1315–1329, aug 2012.

[145] L D'Amiano, D Cozzolino, G Poggi, and L Verdoliva, "Video forgery detection and localization based on 3D patchmatch," in *2015 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. jun 2015, pp. 1–6, IEEE.

[146] Paolo Bestagini, Simone Milani, Marco Tagliasacchi, and Stefano Tubaro, "Codec and GOP Identification in Double Compressed Videos," *IEEE Transactions on Image Processing*, vol. 25, no. 5, pp. 2298–2310, may 2016.

[147] L Kennedy and S.-F. Chang, "Internet image archaeology: automatically tracing the manipulation history of photographs on the web," in *{ACM} International Conference on Multimedia (ACM-MM)*, 2008.

[148] John R Kender, Matthew L Hill, Apostol (Paul) Natsev, John R Smith, and Lexing Xie, "Video genetics," in *Proceedings of the international conference on Multimedia - MM '10*, New York, New York, USA, 2010, p. 1253, ACM Press.

[149] Zanoni Dias, Anderson Rocha, and Siome Goldenstein, "Video Phylogeny: Recovering near-duplicate video relationships," in *2011 IEEE International Workshop on Information Forensics and Security*. nov 2011, pp. 1–6, IEEE.

[150] F O Costa, S Lameri, P Bestagini, Z Dias, A Rocha, M Tagliasacchi, and S Tubaro, "Phylogeny reconstruction for misaligned and compressed video sequences," in *2015 IEEE International Conference on Image Processing (ICIP)*. sep 2015, pp. 301–305, IEEE.

[151] Filipe de O. Costa, Silvia Lameri, Paolo Bestagini, Zanoni Dias, Stefano Tubaro, and Anderson Rocha, "Hash-based frame selection for video phylogeny," in *2016 IEEE International Workshop on Information Forensics and Security (WIFS)*. dec 2016, pp. 1–6, IEEE.

[152] S Lameri, P Bestagini, A. Mellon, S Milani, A Rocha, M Tagliasacchi, and S Tubaro, "Who is my parent? Reconstructing video sequences from partially matching shots," in *2014 IEEE International Conference on Image Processing (ICIP)*. oct 2014, pp. 5342–5346, IEEE.

[153] Alejandro Jaimes, Shih-Fu Chang, and Alexander C Loui, "Duplicate detection in consumer photography and news video," in *Proceedings of the tenth ACM international conference on Multimedia - MULTIMEDIA '02*, New York, New York, USA, 2002, p. 423, ACM Press.

[154] Baris Coskun, Bulent Sankur, and Nasir Memon, "Spatio-Temporal Transform Based Video Hashing," *IEEE Transactions on Multimedia*, vol. 8, no. 6, pp. 1190–1208, dec 2006.

[155] A Melloni, S Lameri, P Bestagini, M Tagliasacchi, and S Tubaro, "Near-duplicate detection and alignment for multi-view videos," in *2015 IEEE International Conference on Image Processing (ICIP)*. sep 2015, pp. 2444–2448, IEEE.

[156] Mo Chen, Jessica Fridrich, Miroslav Goljan, and Jan Lukáš, "Source digital camcorder identification using sensor photo response non-uniformity," in *SPIE Electronic Imaging (EI)*, feb 2007, p. 65051G.

[157] Sevinc Bayram, Husrev Taha Sencar, and Nasir Memon, "Video copy detection based on source device characteristics," in *Proceeding of the 1st ACM international conference on Multimedia information retrieval - MIR '08*, New York, New York, USA, 2008, p. 435, ACM Press.

[158] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool, "Speeded-Up Robust Features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, jun 2008.

[159] Martin A Fischler and Robert C Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, jun 1981.

[160] William M Rand, "Objective Criteria for the Evaluation of Clustering Methods," *Journal of the American Statistical Association*, vol. 66, no. 336, pp. 846, dec 1971.

[161] Paul Jaccard, "The distribution of the flora in the alpine zone," *New Phytologist*, vol. 11, no. 2, pp. 37–50, feb 1912.

[162] C Van Rijsbergen, *Information Retrieval*, Butterworth-Heinemann, 1979.

[163] David L Wallace, E B Fowlkes, and C L Mallows, "A Method for Comparing Two Hierarchical Clusterings: Comment," *Journal of the American Statistical Association*, vol. 78, pp. 553–569, 1983.

[164] Alexander Strehl and Joydeep Ghosh, "Cluster ensembles—a knowledge reuse framework for combining multiple partitions," *Journal of machine learning research*, vol. 3, pp. 583–617, 2002.

[165] Chunhua Chen, Yun Q Shi, and Wei Su, "A machine learning based scheme for double JPEG compression detection," in *2008 19th International Conference on Pattern Recognition*. dec 2008, pp. 1–4, IEEE.

[166] Alin C Popescu and Hany Farid, "Statistical Tools for Digital Forensics," in *International Conference on Information Hiding*, 2004, pp. 128–147.

[167] Tomas Pevny and Jessica Fridrich, "Detection of Double-Compression in JPEG Images for Applications in Steganography," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 2, pp. 247–258, jun 2008.

[168] Bin Li, Yun Q. Shi, and Jiwu Huang, "Detecting doubly compressed JPEG images by using Mode Based First Digit Features," in *2008 IEEE 10th Workshop on Multimedia Signal Processing*. oct 2008, pp. 730–735, IEEE.

[169] Pawel Korus and Jiwu Huang, "Multi-Scale Fusion for Improved Localization of Malicious Tampering in Digital Images," *IEEE Transactions on Image Processing*, vol. 25, no. 3, pp. 1312–1326, mar 2016.

[170] Ali Taimori, Farbod Razzazi, Alireza Behrad, Ali Ahmadi, and Massoud Babaie-Zadeh, "Quantization-Unaware Double JPEG Compression Detection," *Journal of Mathematical Imaging and Vision*, vol. 54, no. 3, pp. 269–286, mar 2016.

[171] Cecilia Pasquini, Giulia Boato, and Fernando Perez-Gonzalez, "Multiple JPEG compression detection by means of Benford-Fourier coefficients," in *2014 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE, dec 2014, pp. 113–118, IEEE.

[172] Cecilia Pasquini, Pascal Schöttle, Rainer Böhme, Giulia Boato, and Fernando Pèrez-Gonzàlez, "Forensics of High Quality and Nearly Identical JPEG Image Recompression," in *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security - IH&MMSec '16*, New York, New York, USA, 2016, pp. 11–21, ACM Press.

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

[174] Irene Amerini, Rudy Becarelli, Roberto Caldelli, and Andrea Del Mastio, "Splicing forgeries localization through the use of first digit features," in *2014 IEEE International Workshop on Information Forensics and Security (WIFS)*. dec 2014, pp. 143–148, IEEE.

[175] Qing Wang and Rong Zhang, "Double JPEG compression forensics based on a convolutional neural network," *EURASIP Journal on Information Security*, vol. 2016, no. 1, pp. 23, dec 2016.

[176] Weiqi Luo, Zhenhua Qu, Jiwu Huang, and Guoping Qiu, "A Novel Method for Detecting Cropped and Recompressed Image Block," in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*. 2007, pp. II–217–II–220, IEEE.

[177] Yi-Lei Chen and Chiou-Ting Hsu, "Detecting Recompression of JPEG Images via Periodicity Analysis of Compression Artifacts for Tampering Detection," *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 2, pp. 396–406, jun 2011.

[178] Zhenhua Qu, Weiqi Luo, and Jiwu Huang, "A convolutive mixing model for shifted double JPEG compression with application to passive image authentication," in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*. mar 2008, pp. 1661–1664, IEEE.

[179] Tiziano Bianchi and Alessandro Piva, "Detection of Nonaligned Double JPEG Compression Based on Integer Periodicity Maps," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 842–848, apr 2012.

[180] Tiziano Bianchi and Alessandro Piva, "Image Forgery Localization via Block-Grained Analysis of JPEG Artifacts," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 1003–1017, jun 2012.

[181] Duc-Tien Dang-Nguyen, Cecilia Pasquini, Valentina Conotter, and Giulia Boato, "RAISE: a raw images dataset for digital image forensics," in *Proceedings of the 6th ACM Multimedia Systems Conference on - MMSys '15*, New York, New York, USA, 2015, pp. 219–224, ACM Press.

[182] Xavier Glorot and Yoshua Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

[183] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell, "Caffe: Convolutional Architecture for Fast Feature Embedding," *arXiv preprint arXiv:1408.5093*, jun 2014.

[184] Jianquan Yang, Jin Xie, Guopu Zhu, Sam Kwong, and Yun-Qing Shi, "An Effective Method for Detecting Double JPEG Compression With the Same Quantization Matrix," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 11, pp. 1933–1942, nov 2014.

[185] André Braz, Maria López-López, and Carmen García-Ruiz, "Raman spectroscopy for forensic analysis of inks in questioned documents," *Forensic Science International*, vol. 232, no. 1-3, pp. 206–212, oct 2013.

[186] Po-Chun Chu, Bruno Yue Cai, Yeuk Ki Tsoi, Ronald Yuen, Kelvin S.Y. Leung, and Nai-Ho Cheung, "Forensic Analysis of Laser Printed Ink by X-ray Fluorescence and Laser-Excited Plume Fluorescence," *Analytical Chemistry*, vol. 85, no. 9, pp. 4311–4315, may 2013.

[187] Pei-Ju Chiang, N Khanna, A. Mikkilineni, M.V.O. Segovia, Sungjoo Suh, J. Allebach, G. Chiu, and E. Delp, "Printer and scanner forensics," *IEEE Signal Processing Magazine*, vol. 26, no. 2, pp. 72–83, mar 2009.

[188] Gazi N Ali, Pei-Ju Chiang, Aravind K Mikkilineni, Jan P Allebach, George T.-C. Chiu, and Edward J Delp, "Intrinsic and Extrinsic Signatures for Information Hiding and Secure Printing with Electrophotographic Devices," in *International Conference on Digital Printing Technologies*, sep 2003, pp. 511–515.

[189] Iuliia Tkachenko, William Puech, Christophe Destruel, Olivier Strauss, Jean-Marc Gaudin, and Christian Guichard, "Two-Level QR Code for Private Message Sharing and Document Authentication," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 3, pp. 571–583, mar 2016.

[190] Matthew D Gaubatz and Steven J Simske, "Printer-scanner identification via analysis of structured security deterrents," in *2009 First IEEE International Workshop on Information Forensics and Security (WIFS)*. dec 2009, pp. 151–155, IEEE.

[191] Steven J Simske, Jason S Aronoff, Margaret Sturgill, and Juan Carlos Villa, "Spectral Pre-Compensation and Security Print Deterrent Authentication," *NIP and Digital Fabrication Conference*, vol. 2008, no. 2, pp. 792–795, 2008.

[192] Joe Frost, "The Real Reason You Can't Print Without Colour Ink: Printers Leave Tiny Yellow Dots for Authorities to Identify Documents," http://www.techly.com.au/2015/09/24/real-reason-cant-print-without-colour-ink-printers-leave-tiny-yellow-dots-authorities-identify-documents/, sep 2015.

[193] Aravind K Mikkilineni, Pei-ju Chiang, Gazi N Ali, George T c. Chiu, Jan P Allebach, and Edward J Delp, "Printer identification based on textural features," in *International Conference on Digital Printing Technologies*, oct 2004, pp. 306–311.

[194] Eric Kee and Hany Farid, "Printer profiling for forensics and ballistics," in *Proceedings of the 10th ACM workshop on Multimedia and security - MM&Sec '08*, New York, New York, USA, sep 2008, p. 3, ACM Press.

[195] Anselmo Ferreira, Luiz C Navarro, Giuliano Pinheiro, Jefersson A. dos Santos, and Anderson Rocha, "Laser printer attribution: Exploring new features and beyond," *Forensic Science International*, vol. 247, pp. 105–125, feb 2015.

[196] Heung-Kyu Lee and Do-Guk Kim, "Colour laser printer identification using halftone texture fingerprint," *Electronics Letters*, vol. 51, no. 13, pp. 981–983, jun 2015.

[197] Aravind K Mikkilineni, Osman Arslan, Pei-ju Chiang, Roy M Kumontoy, Jan P Allebach, and George T c, "Printer forensics using {SVM} techniques," in *International Conference on Digital Printing Technologies*, oct 2005, pp. 223–226.

[198] Min-Jen Tsai, Jin-Shen Yin, Imam Yuadi, and Jung Liu, "Digital forensics of printed source identification for Chinese characters," *Multimedia Tools and Applications*, vol. 73, no. 3, pp. 2129–2155, dec 2014.

[199] Min-Jen Tsai, Chien-Lun Hsu, Jin-Sheng Yin, and Imam Yuadi, "Japanese character based printed source identification," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*. may 2015, pp. 2800–2803, IEEE.

[200] Sara Elkasrawi and Faisal Shafait, "Printer Identification Using Supervised Learning for Document Forgery Detection," in *2014 11th IAPR International Workshop on Document Analysis Systems*. apr 2014, pp. 146–150, IEEE.

[201] Nitin Khanna, Aravind K Mikkilineni, George T. C. Chiu, Jan P Allebach, and Edward J Delp, "Survey of Scanner and Printer Forensics at Purdue University," in *Computational Forensics*, Berlin, Heidelberg, aug 2008, pp. 22–34, Springer Berlin Heidelberg.

[202] Hae-Yeoun Lee and Jung-Ho Choi, "Identifying Color Laser Printer Using Noisy Feature and Support Vector Machine," in *2010 Proceedings of the 5th International Conference on Ubiquitous Information Technologies and Applications*. dec 2010, pp. 1–6, IEEE.

[203] Jung-Ho Choi, Heung-Kyu Lee, Hae-Yeoun Lee, and Young-Ho Suh, "Color laser printer forensics with noise texture analysis," in *Proceedings of the 12th ACM workshop on Multimedia and security - MM&Sec '10*, New York, New York, USA, 2010, p. 19, ACM Press.

[204] Robert M Haralick, K Shanmugam, and Its'Hak Dinstein, "Textural Features for Image Classification," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-3, no. 6, pp. 610–621, nov 1973.

[205] Jung-Ho Choi, Dong-Hyuck Im, Hae-Yeoun Lee, Jun-Taek Oh, Jin-Ho Ryu, and Heung-Kyu Lee, "Color laser printer identification by analyzing statistical features on discrete wavelet transform," in *2009 16th IEEE International Conference on Image Processing (ICIP)*. nov 2009, pp. 1505–1508, IEEE.

[206] Min-Jen Tsai, Jung Liu, Chen-Sheng Wang, and Ching-Hua Chuang, "Source color laser printer identification using discrete wavelet transform and feature selection algorithms," in *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*. may 2011, pp. 2633–2636, IEEE.

[207] Nitin Khanna, George T C Chiu, Jan P Allebach, and Edward J Delp, "Scanner identification with extension to forgery detection," in *Procedures of SPIE*, Edward J. Delp III, Ping Wah Wong, Jana Dittmann, and Nasir D. Memon, Eds., feb 2008, p. 68190G.

[208] Nobuyuki Otsu, "A Threshold Selection Method from Gray-Level Histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, jan 1979.

[209] Orhan Bulan, Junwen Mao, and Gaurav Sharma, "Geometric distortion signatures for printer identification," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*. apr 2009, pp. 1401–1404, IEEE.

[210] Han Wu, Xiangwei Kong, and Shize Shang, "A printer forensics method using halftone dot arrangement model," in *2015 IEEE China Summit and International Conference on Signal and Information Processing (ChinaSIP)*. jul 2015, pp. 861–865, IEEE.

[211] Seung-Jin Ryu, Hae-Yeoun Lee, Dong-Hyuck Im, Jung-Ho Choi, and Heung-Kyu Lee, "Electrophotographic printer identification by halftone texture analysis," in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. mar 2010, pp. 1846–1849, IEEE.

[212] Do-Guk Kim and Heung-Kyu Lee, "Color laser printer identification using photographed halftone images," in *European Signal Processing Conference (EUSIPCO)*, sep 2014, pp. 795–799.

[213] Aravind K Mikkilineni, Pei-ju Chiang, Gazi N Ali, George T. C. Chiu, Jan P Allebach, and Edward J. Delp III, "Printer identification based on graylevel co-occurrence features for security and forensic applications," in *SPIE Security, Steganography, and Watermarking of Multimedia Contents*, Edward J. Delp III and Ping W. Wong, Eds., mar 2005, p. 430.

[214] Pavel Micka, "Letter frequency (English)," \url{http://en.algoritmy.net/article/40379/Letter-frequency-English}.

[215] Aravind K Mikkilineni, Nitin Khanna, and Edward J Delp, "Forensic printer detection using intrinsic signatures," in *SPIE Media Watermarking, Security, and Forensics*, Nasir D. Memon, Jana Dittmann, Adnan M. Alattar, and Edward J. Delp III, Eds., feb 2011, p. 78800R.

[216] Weina Jiang, Anthony T S Ho, Helen Treharne, and Yun Q Shi, "A Novel Multi-size Block Benford's Law Scheme for Printer Identification," in *Pacific Rim Conference on Advances in Multimedia Information Processing*, 2010, pp. 643–652.

[217] Gazi N Ali, Pei-ju Chiang, Aravind K Mikkilineni, George T Chiu, Edward J Delp, and Jan P Allebach, "Application of principal components analysis and gaussian mixture models to printer identification," in *International Conference on Digital Printing Technologies*, oct 2004, pp. 301–305.

[218] Min-Jen Tsai and Jung Liu, "Digital forensics for printed source identification," in *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*. may 2013, pp. 2347–2350, IEEE.

[219] R Duda and P Hart., *Pattern Classification and Scene Analysis*, vol. 44, John Wiley and Sons, jul 1974.

[220] Yubao Wu, Xiangwei Kong, Xin'gang You, and Yiping Guo, "Printer forensics based on page document's geometric distortion," in *2009 16th IEEE International Conference on Image Processing (ICIP)*. nov 2009, pp. 2909–2912, IEEE.

[221] Marco Schreyer, Christian Schulze, Armin Stahl, and Wolfgang Effelsberg, "Intelligent Printing Technique Recognition and Photocopy Detection for Forensic Document Examination," in *Informatiktage*, mar 2009, pp. 39–42.

[222] A Vedaldi and K Lenc, "MatConvNet – Convolutional Neural Networks for MATLAB," in *Proceeding of the {ACM} Int. Conf. on Multimedia*, 2015.

[223] Cees G M Snoek, Marcel Worring, and Arnold W. M. Smeulders, "Early versus late fusion in semantic video analysis," in *Proceedings of the 13th annual ACM international conference on Multimedia - MULTIMEDIA '05*, New York, New York, USA, nov 2005, p. 399, ACM Press.

[224] Anderson Rocha and Siome Klein Goldenstein, "Multiclass From Binary: Expanding One-Versus-All, One-Versus-One and ECOC-Based Approaches," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 2, pp. 289–302, feb 2014.

[225] Thomas G. Dietterich, "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms.," *Neural computation*, vol. 10, no. 7, pp. 1895–1923, sep 1998.

[226] Norbert Wiener, *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*, The MIT Press, 1964.

[227] Timo Ojala, Matti Pietikäinen, and David Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern Recognition*, vol. 29, no. 1, pp. 51–59, jan 1996.

[228] Navneet Dalal and Bill Triggs, "Histograms of Oriented Gradients for Human Detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. 2005, vol. 1, pp. 886–893, IEEE.

[229] Rafael C Gonzalez and Richard E Woods, *Digital Image Processing*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 2001.

[230] Frédo Durand and Julie Dorsey, "Fast bilateral filtering for the display of high-dynamic-range images," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 257–266, jul 2002.

[231] Kaiming He, Jian Sun, and Xiaoou Tang, "Guided Image Filtering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 6, pp. 1397–1409, jun 2013.