Politecnico di Milano
Dipartimento di Elettronica, Informazione e Bioingegneria
Doctoral Programme In Information Technology

Machine Learning techniques for integrity control

Doctoral Dissertation of:
Nicolò Bonettini

Supervisor:
Prof. Marco Marcon

Tutor:
Prof. Andrea Virgilio Monti Guarnieri

The Chair of the Doctoral Program:
Prof. Luigi Piroddi

2021 – XXXIV Cycle

Abstract

THis thesis focuses on two particular aspects of integrity control: *digital integrity* and *physical integrity*. In particular, we propose Signal Processing, Machine Learning and Deep Learning techniques for solving relevant problems in these two areas. Given the vastness of these fields, we select two applications we consider the most relevant to investigate: Multimedia Forensics for the *digital integrity* world, and Food Safety Hyperspectral X-ray analysis for the *physical integrity* world.

Digital integrity covers the crucial role of verifying and assessing trustworthiness in digital media content. We spend more and more time on social networks and chats, where we are constantly flooded with images and videos. Therefore, we strongly need tools to ensure that those contents are not tampered with. The more we continue in this vein, the more we are exposed to the risk of being fooled or manipulated by multimedia content over the internet. To solve this issue, we investigate three specific areas of interest: sensor integrity, coding integrity and semantic integrity. Sensor integrity deals with the integrity of the traces left by camera sensors on digital images. We know from the literature that it is possible to exploit sensor traces to bind a picture to the device that shot it effectively. In this work, to study the robustness of sensor-based integrity methods, we focus on image "anonymization", i.e., the possibility of making device attribution techniques fail. We first propose a method for accomplishing image anonymization, and then a method for detecting it, both based on Convolutional Neural Networks.

Coding integrity is about the integrity of the traces left by coding steps in the de-facto image compression standard format JPEG. This integrity is essential, as tampering with a JPEG image followed by another JPEG compression would likely compromise the typical structure left by the first coding step. We deeply investigate this scenario, proposing a method for detecting double JPEG compression, a method for detecting multiple compressions up to four steps, and a method for detecting double compression performed with different implementations of the codec.

Semantic integrity regards the integrity of the meaning of images and videos, i.e., what we semantically perceive by watching them. More and more techniques allow the fully automatic generation of very realistic synthetic images and videos with minimal effort and no artistic nor graphic knowledge required. Consequently, we constantly face dramatic problems, like impersonation or puppeteering of someone in a video for malicious intent. We propose a method for detecting images generated through Generative Adversarial Networks and a method for detecting videos generated by Deepfake neural networks.

Physical integrity concerns the integrity of objects: a gigantic field spanning from controlling buildings and infrastructures' integrity to checking integrity in microscopic manufacturing. Among all the possible applications in this broad field, we consider the application of Hyperspectral X-ray analysis in the context of Food Safety, that is, using this powerful method to find contaminants buried in food in different stages of its industrial preparation. In this sense, we cover the integrity of the acquired signal, which is likely to be compromised whether a foreign body alters its characteristics. Compared to traditional X-ray analysis, the Hyperspectral allows finer scans of the object at hand. The whole radiation energy is divided into multiple sub-bands that we can analyze separately. This makes it possible to detect low-density contaminants (e.g., plastic polymers) undetectable by single-energy or dual-energy X-ray analysis. Unfortunately, this kind of X-ray acquisition suffers from noise due to the probabilistic nature of photons that cross the material and are counted by the sensor.

For this reason, we start by investigating the task of denoising Hyperspectral X-ray acquisitions as a required step for performing other kinds of processing in a second moment. We first propose a data-driven method based on a Convolutional AutoEncoder, comparing it with a more traditional model-based Wiener filter approach. Continuing in this vein, we explore the world of AutoEncoders for the denoising task. We propose a comparison between different AutoEncoders variants to exploit physical modeling of dimensionality reduction. Finally, we move to the more application-oriented task of plastic polymers classification in industrial food preparation pipelines. We propose a method based on a Convolutional AutoEncoder to estimate the physical parameters related to plastic polymers and low-density metals. The proposed solution implements the multitask-learning paradigm to jointly denoise the X-ray acquisition provided as input while performing the parameter estimation.

The results achieved in this thesis show that a mix of Signal Processing and Machine Learning is beneficial to solve a whole variety of integrity-related problems. Whenever a purely model-based solution proves lacking in some respect (e.g., due to simplistic assumptions), data-driven methods work as an alternative viable solution. Conversely, when a model for the problem at hand is well defined (e.g., due to physical constraints), signal processing may greatly help purely-data driven methods.

Contents

List of notation

x	Scalar value
$\mathbf{x}$	Column vector
x_i	i-th element of vector $\mathbf{x}$
$[\mathbf{x}]_i$	i-th element of vector $\mathbf{x}$
$\hat{\mathbf{x}}$	Estimation of $\mathbf{x}$
$\alpha \cdot \mathbf{x}$	Scalar multiplication
$\mathbf{X}$	Matrix
$\mathbf{X} \circ \mathbf{Y}$	Hadamard product
$\mathbf{XY}$	Matrix multiplication
$A(\cdot)$	Operator
$\mathscr{L}(\cdot)$	Loss function
$\mathcal{S}$	Set
M	Metric

CHAPTER 1

Introduction

According to the Oxford Dictionary, the word “integrity” comes from the latin word *integer*, composed by the negation particle *in* and the root of latin word *tangere*, “to touch”[1]. Similarly, the word “intact” comes from *in* + *tact*, the past participle of *tangere*[2]. The original meaning of the word “integrity” is, therefore “being untouched”, “being intact”, “not being compromised or tampered with”.

We value integrity in everyday life. We constantly seek pristine conditions to assess the trustworthiness of something. Industries perform all kinds of integrity checks during the production process to ensure the best quality of their products. Thinking about our daily life, we want the food we buy at the supermarket to be immaculate as a warranty of no contamination in the supply chain. Similarly, when we see a photograph posted on some online newspaper, we implicitly would like the picture to be real, not tampered with to accomplish malicious intentions. This holds in many practical aspects, even if we are not explicitly thinking about it.

Determining and measuring the amount of integrity of something is no easy task. We need to define what we mean with integrity for a particular object. The answer seems evident for tangible goods: an object is intact when we cannot perceive physical damage. A vegetable is intact if it is not rotten. A piece of furniture is intact if it is not broken, and so on. However, measuring this integrity can be more challenging than it looks like. For example, the two objects we just mentioned could appear perfect and intact from the outside but have troubles inside. The vegetable could be rotten from the heart. The piece of furniture could have structural weaknesses we cannot detect from an outer look. Things get even more complicated if we start thinking about intangible assets, like digital multimedia content. All sorts of digital content surround us: social

[1] `https://www.lexico.com/definition/integrity`
[2] `https://www.lexico.com/definition/intact`

media images, videos, vocal messages from chats, music from streaming platforms. How can we assess the integrity of digital content? What do we care about? The perceived quality? The semantic? Of course, there is not a unique answer to this question. Generally speaking, we can call digital content intact if it has not been tampered with to change its original meaning, the purpose it was created for. For example, a video of politicians speaking can be doctored to make them say something completely out of their thoughts, which can have a relevant impact on society, making the video not intact anymore.

This thesis analyzes two particular integrity control applications out of the many possibilities. The first one is digital Multimedia Forensics, which is a good example of the *digital integrity* world. The second one is Hyperspectral X-ray analysis for Food Safety, which belongs to the world of *physical integrity* of tangible goods.

While these two tasks might seem to come from two unrelated worlds, there are many contact points between the two. Industries usually employ different kinds of quality control on their product on the conveyor belt, from classic visual control through pictures and videos to deep inspection with X-rays. Besides, having tools to determine *digital integrity* can be crucial. For example, if a client files a complaint about damaged goods providing pictures of them, these pictures could have been doctored to obtain a refund or some sort of benefit maliciously.

By exploiting the common point and diversity of these two worlds, we aim to have a better perspective on the integrity control problem. In our approach, we enforce classic signal processing techniques to define discriminative features to feed data-driven models. This allows easing the training of models that would otherwise require massive data, making their applications possible even in data-scarcity or low-computational scenarios.

In this chapter, we first summarize the solutions and advancements proposed in this thesis. Then, we provide the outline of the thesis and finally we list the peer-reviewed conference and journal papers we included in this thesis.

1.1 *Digital Integrity* - Multimedia Forensics

In the last decade, the massive adoption of the smartphone and the global popularization of social networking websites have led to unprecedented rates of social digital media sharing. In 2010 alone, Facebook reported storing more than 260 billion images, with users uploading one billion new images each week [1]. In 2016, Instagram had more than 400 million active monthly users who shared over 40 billion images, with an average of 3.5 billion daily “likes” for more than 80 million images shared daily on the site [2].

In the fast-paced world we live in, technology is becoming more and more complex and immersive, surrounding us even when we are not consciously aware of it. The advent of social media has shifted the human communication paradigm in the last decade. We are constantly flooded with newsfeeds, status updates, holiday pictures, funny videos of strangers doing all the kinds of awesome stuff. We are so overwhelmed by social media content that we cannot devote proper attention to all of them. The constant exposition lowers our defence against multimedia counterfeits, which are created with the precise and subtle purpose of changing our minds about a specific subject.

As a matter of fact, in the last few years, the fake news plague has reached unseen levels in human history. According to the 2019 global inventory of organised social media manipulation [3], social media have been used for organised manipulation campaigns in more than 40 countries in 2018 only. Computational propaganda, namely "the use of algorithms, automation, and big data to shape public life" [3], is becoming mainstream for influencing global audiences with social and political implications. Experts are worried about future political elections, which are likely going to be highly influenced by all the kinds of visual content circulating over the Internet, and the menace is far from over [4, 5].

Fortunately, the increasing amount of data we keep producing every day also offers the great opportunity to develop data-driven integrity control techniques. Machine Learning and Deep Learning models can be successfully employed to find discriminative patterns and detect whether some tampering has occurred. Developing this kind of techniques is the role of Multimedia Forensics researchers, and one of the goals of this thesis.

Multimedia Forensics is the discipline that finds alterations in digital multimedia content. We focus on three particular topics of Multimedia Forensics. The first one is the problem of Source Device Anonymization and its detection. Every phone or digital camera leaves traces on the acquired pictures, which bind the image to the device that shot it. There are techniques to hide these traces and make the binding process impossible, thus anonymizing the image. By detecting image anonymization, we can assess if someone tampered with the image. The second topic is about detecting multiple JPEG compressions on an image, which can again be a hint for detecting alterations. The third topic is detecting synthetically generated images and videos, which is a scorching topic in the Multimedia Forensics community. Given the recent advances of generative Deep Learning models, very realistic images and videos of persons, animals and objects can be created without particular technical abilities. Detecting them is a crucial waypoint to fight identity stealing, revenge porn and other widespread malicious practices over the internet.

1.2 *Physical Integrity* - Hyperspectral X-ray for Food Safety

Food Safety is a discipline that oversees the preparation, handling and storage of food in order to prevent food-borne illness. Food contamination can happen during different stages of its preparation cycle: cooking, packaging, storing, and transportation. For this reason, the European Commission has defined a series of food safety guidelines inspired by Hazard Analysis and Critical Control Points (HACCP) standards [6]. These impose standard food inspection rules and regulations on all producers.

There are many techniques for inspecting food during the various phases of preparation. Some of them are intrusive, requiring physical access to the food, and some are non-intrusive.

In this thesis we focus on *physical integrity* by means of Hyperspectral X-ray imaging [7], a non-intrusive technique that enables the analysis of the interior part of the object at hand. The main advantage of this technique is its ability to detect low-density contaminants in food, such as plastic polymers and materials with a similar density to organic compounds. Unfortunately, the signal acquired from the sensor is very noisy, and a specific denoising pipeline is required to perform real-time analysis on production sites. In this thesis, we propose a comparison between different denoising algorithms, both model-based and data-driven. We show how Deep Learning techniques can be effectively used to reduce the noise before further analysing the signal. We also propose a method for classifying plastic polymers and metals of different densities and thicknesses while retaining a denoising pipeline that can be further used for synthesis purposes.

1.3 Original contributions

This thesis proposes advances in various aspects of integrity control, ranging from Multimedia Forensics to Food Safety.

The problem of source device anonymization has been considered both from the forensics and the counterforensics viewpoint. We propose a solution to anonymize an image while retaining its visual quality and a method for detecting the image anonymization as a hint of severe tampering with. Regarding the JPEG forensics, we propose a method for detecting double compression both in the case of aligned and non-aligned second coding step. We also consider the multiple compression case up to four steps, and the case in which the second compression is performed with a different implementation with respect to the first. We also study the problem of detecting images and videos generated by Deep Learning methods, a very hot topic in today forensics literature, proposing one detection method apiece.

As far as Food Safety is concerned, we consider the application of contaminant detection by means of Hyperspectral X-ray analysis. A common problem in treating this kind of signals is dealing with the sensor noise, which greatly affects the acquisitions. We first propose a comparative study between a statistical denoising method based on Wiener filter and a Deep Learning method. We then consider the task of classifying plastic polymers, which are typically found as contaminants in industrial food. By doing so in a multitask fashion, we propose a solution that jointly performs a denoising step and estimates the physical parameters associated with the polymer, by employing a Convolutional AutoEncoder (CAE). Intrigued by the possibilities offered by this architecture, we finally propose a comparative study on different AutoEncoder (AE) variants, for the pure task of acquisition denoising.

In this section we provide a concise description of every original contribution that we developed and published (or submitted for publication) in peer-reviewed conferences and journals.

1.3.1 A method for image anonymization

In the last few years, forensic researchers have developed a wide set of techniques to blindly attribute an image to the device used to shoot it. Among these techniques, those based on Photo Response Non Uniformity (PRNU) [8] have shown incredibly accurate results, thus they are often considered as a reference baseline solution. The rationale behind these techniques is that each camera sensor leaves on acquired images a characteristic noise pattern. This pattern can be estimated and uniquely mapped to a specific acquisition device through a cross-correlation test [9]. We study the possibility of leveraging recent findings in the Deep Learning field to attack PRNU-based detectors. Specifically, we focus on the possibility of editing an image through Convolutional

Neural Networks (CNNs) in a visually imperceptible way, still hindering PRNU noise estimation. Results show that performing such an attack is possible, even though an informed forensic analyst can reduce its impact through a smart test.

1.3.2 Detection of image anonymization

It is of great importance to develop methods to detect image anonymization, to keep the pace with the latest anonymization methods [10]. We present an approach to detect whether an image has undergone a laundering process, i.e., it has been tampered with so that its unique characterizing features have been changed to avoid detection. We focus on PRNU noise unique to every imaging sensor, we consider that an image has been "laundered" when we detect the absence of PRNU from an image.

We propose a per image preprocessing pipeline that generates information-rich features later used as input of fine-tuned CNNs. We study the performance of the proposed approach using various CNN architectures and blind anonymization techniques, and show its effectiveness under several training and testing scenarios. Our results also show that CNN models trained with the proposed feature are capable of generalizing over unseen devices and are robust against non-geometric transformations.

1.3.3 Aligned and non-aligned double JPEG detection

Due to the wide diffusion of JPEG coding standard, the image forensic community has devoted significant attention to the development of Double JPEG (DJPEG) compression detectors through the years [11–13]. The ability of detecting whether an image has been compressed twice provides paramount information toward image authenticity assessment. Given the trend recently gained by CNNs in many computer vision tasks, we propose to use CNNs for aligned and non-aligned double JPEG compression detection. In particular, we explore the capability of CNNs to capture DJPEG artifacts directly from images. Results show that the proposed CNN-based detectors achieve good performance even with small size images (i.e., 64×64), outperforming state-of-the-art solutions, especially in the non-aligned case. Besides, good results are also achieved in the commonly-recognized challenging case in which the first Quality Factor (QF) is larger than the second one.

1.3.4 Multiple JPEG compression detection by means of non-negative matrix factorization

JPEG compression is typically operated first at image inception time directly on the acquisition device. Then, it is customary re-applied every time an image is manipulated or shared through social media [14]. For this reason, the more the applied JPEG compressions, the higher the likelihood that an image underwent some editing. For this reason, we propose an algorithm to detect multiple JPEG compressions, specifically up to four coding cycles. This approach leverages the Task-driven Non-negative Matrix Factorization (TNMF) model, fed with histograms of the Discrete Cosine Transform (DCT) of the image under analysis. Experimental results show the effectiveness of the method if compared with the state-of-the-art, confirming this strategy as a viable solution for detecting multiple JPEG compressions.

1.3.5 Different JPEG implementation detection based on eigen-algorithms

The JPEG standard does not define a unique rule regarding the quantization of the DCT coefficients [15]. Every implementer is free to adopt his own quantization rule, thus every implementation leave its own traces. We focus on the detection of traces left on images by the use of different JPEG implementations (e.g., characteristic of proprietary software suites). Specifically, given a JPEG image under analysis, we propose a compact descriptor that enables to distinguish which JPEG implementation has been used. This is done considering the challenging scenario in which the same quantization matrix is used. Results show that it is possible to distinguish between two popular JPEG implementations, as well as to adapt the proposed methodology to additional JPEG-based forensics problems.

1.3.6 Detection of images generated by Generative Adversarial Networks

The advent of Generative Adversarial Network (GAN) architectures has given anyone the ability of generating incredibly realistic synthetic imagery [16]. The malicious diffusion of GAN-generated images may lead to serious social and political consequences (e.g., fake news spreading, opinion formation, etc.). It is therefore important to regulate the widespread distribution of synthetic imagery by developing solutions able to detect them. We study the possibility of using Benford's law to discriminate GAN-generated images from natural photographs. Benford's law describes the distribution of the most significant digit for quantized DCT coefficients. Extending and generalizing this property, we show that it is possible to extract a compact feature vector from an image. This feature vector can be fed to an extremely simple classifier for GAN-generated image detection purpose.

1.3.7 Detection of generated Deepfake videos

In the last few years, several techniques for facial manipulation in videos have been successfully developed and made available to the masses (i.e., FaceSwap [17], Deepfakes, etc.). These methods enable anyone to easily edit faces in video sequences with incredibly realistic results and a very little effort. Despite the usefulness of these tools in many fields, if used maliciously, they can have a disastrous impact on society (e.g., fake news spreading, cyberbullying through fake revenge porn). The ability of objectively detecting whether a face has been manipulated in a video sequence is then a task of utmost importance. We tackle the problem of face manipulation detection in video sequences targeting modern facial manipulation techniques. In particular, we study the ensembling of different trained CNN models. In the proposed solution, different models are obtained starting from a base network (i.e., EfficientNetB4 [18]) making use of two different concepts: (i) attention layers; (ii) siamese training. We show that combining these networks leads to promising face manipulation detection results on two publicly available datasets with more than 119000 videos.

1.3.8 Denoising techniques for Hyperspectral X-ray acquisitions

Hyperspectral X-ray analysis is a powerful method for detecting low-density contaminants in food. Unfortunately, the acquired signal is greatly affected by noise, making the detection of the contaminant more challenging. Therefore, a good denoising pipeline is paramount to perform real-time detection of foreign bodies. In particular, we compare a classical model-based Wiener filter solution with a data-driven methodology based on a CAE. A challenging aspect is related to the specific kind of 2D signal we are processing: it presents mixed dimensions information since on the vertical axis there is the pixels position while, on the abscissa, there are the different energy bins associated to the acquired X-ray spectrum. The goal is to approximate the denoising function using a learning-from-data approach and to verify its capability to emulate the Wiener filter using a much less demanding approach in terms of signal and noise statistical knowledge. We show that, after training, the CNN is able to properly restore the 2D signal with results very close to the Wiener filter, honouring the proper signal shape.

1.3.9 Comparing different AutoEncoder variants for Hyperspectral X-ray denoising

We continue studying the problem of hyperspectral X-ray image denoising. We propose a comparison between three different AE variants: the Variational AutoEncoder (VAE), the AE, and the Augmented AutoEncoder (AuAE) itself. All the networks are trained in an unsupervised fashion to denoise a given noisy spectrum. We force the latent space of the networks to have just two parameters, as suggested by Lambert-Beer physical law [19, 20]. We validate our experiments on a synthetic dataset composed of roughly 15 million spectra. Results suggest that the Augmented Autoencoder is the best network in this task, showing excellent performance without suffering from the non-deterministic behaviour of the VAE.

1.3.10 A multitask approach for denoising and classification of Hyperspectral X-ray acquisitions

X-ray acquisitions are beneficial in food contaminant analysis as they can detect both metallic and non-metallic objects. We consider the scenario of single-pixel hyperspectral X-ray acquisitions applied to a series of materials with different characteristics. We propose a method that jointly applies a denoising operation and detects the analyzed material in terms of a physical parameterization. The proposed algorithm is based on a CNN trained with a multi-task learning strategy using a custom loss function tailored to the problem at hand. Experimental results on metals and polymers show that the proposed method can also generalize to materials never seen at training time.

1.4 Outline

To ease the readability of this thesis, our contribution has been divided into two parts and eight chapters.

Part I concerns the *digital integrity* world.

Chapter 2 introduces the sensor integrity problem, that is the integrity of the traces left by the digital camera on images. Section 2.1 presents the state of the art of image source attribution. Section 2.2 provides a background on the required concepts we adopt in the chapter. Section 2.3 describes the proposed solution [21] to the image anonymization problem. Section 2.4 describes the proposed solution [22] for exploiting the traces left by anonymization techniques to detect anonymization. Section 2.5 draws some common conclusions on the proposed methods and concludes the chapter.

Chapter 3 treats the integrity of the traces left by the JPEG compression algorithm on images. Section 3.1 provides the state of the art of JPEG forensics. Section 3.2 introduces background concepts on the JPEG compression algorithm. Section 3.3 describes the proposed method [23] for detecting double JPEG compression in the two possible scenarios of aligned and non-aligned second compression. Section 3.4 describes the proposed method [24] for detecting up to four JPEG coding steps. Section 3.5 describes the proposed method [25] for detecting double compression with a different implementation of the JPEG algorithm. Section 3.6 draws a common conclusion on the three proposed methods, concluding the chapter.

Chapter 4 is about the semantic integrity of images and videos, i.e., the meaning we perceive by watching them. Section 4.1 provides the state of the art for the detection of synthetically generated images and videos. Section 4.2 introduces the main concepts used in the proposed methods. Section 4.3 describes the proposed method [26] about the detection of GAN generated images. Section 4.4 describes the proposed method [27] about the detection of Deepfakes videos. Section 4.5 draws common conclusion on the chapter.

Part II concerns the *physical integrity* world.

Chapter 5 introduces the Hyperspectral X-ray analysis in food safety applications. This introductory chapter is organized as follows: Section 5.1 provides some state of the art for the use of X-ray analysis in food safety. Section 5.2 describes the X-ray acquisition system we consider to obtain the hyperspectral acquisitions used in the coming chapters. Section 5.3 provides some background on the Lambert-Beer law, a physical law we exploit in the next chapters. Section 5.4 concludes the chapter.

Chapter 6 treats the problem of denoising Hyperspectral X-ray acquisitions. Section 6.1 gives important details on the concepts of Wiener filter and AE that we use throughout the chapter. Section 6.2 describes the proposed method [28] on the comparison between a model-based and a data-driven technique for denoising. Section 6.3 describes the proposed method [] on the use of AE variants for the denoising task. Section 6.4 draws common conclusions between the two proposed methods and concludes the chapter.

Chapter 7 treats the problem of polymers classification with Hyperspectral X-ray. Section 7.1 [29] describes the proposed method for plastic polymers classification. Section 7.2 describes the experimental campaign we designed for validating our method. Section 7.3 provides an overview on the experimental results. Section 7.4 draws conclusions on the chapter.

Finally, Chapter 8 draws final considerations on the work presented in this thesis and proposes some future research lines on the considered topics.

1.5 List of included publications

In this section we provide the complete list of articles published (or submitted to) international conferences and journals during the Ph.D. period.

- [23] M. Barni, L. Bondi, N. Bonettini, P. Bestagini, A. Costanzo, M. Maggini, B. Tondi, S. Tubaro, “Aligned and Non-Aligned Double JPEG Detection Using CNNs”, *Journal of Visual Communication and Image Representation (JVCI)*, 2017
- [24] S. Mandelli, N. Bonettini, P. Bestagini, V. Lipari, S. Tubaro, “Multiple JPEG compression detection through task-driven non-negative matrix factorization”, *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, Canada, 2018
- [21] N. Bonettini, L. Bondi, D. Güera, S. Mandelli, P. Bestagini, S. Tubaro, E. J. Delp, “Fooling PRNU-Based Detectors Through CNNs”, *European Signal Processing Conference (EUSIPCO)*, Rome, Italy, 2018
- [25] N. Bonettini, L. Bondi, P. Bestagini, S. Tubaro, “JPEG Implementation Forensics Based on Eigen-Algorithms”, *International Workshop on Information Forensics and Security (WIFS)*, Hong Kong, 2018
- [22] N. Bonettini, D. Güera, L. Bondi, P. Bestagini, E. J. Delp, S. Tubaro, “Image Anonymization Detection With Deep Handcrafted Features”, *IEEE International Conference on Image Processing (ICIP)*, Taipei, Taiwan, 2019
- [28] N. Bonettini, M. Paracchini, P. Bestagini, M. Marcon, S. Tubaro, “Hyperspectral X-ray Denoising: Model-Based and Data-Driven Solutions”, *European Signal Processing Conference (EUSIPCO), La Coruna, Spain*, 2019
- [27] N. Bonettini, E.D. Cannas, S. Mandelli, L. Bondi, P. Bestagini, S. Tubaro, “Video Face Manipulation Detection Through Ensemble of CNNs”, *IAPR International Conference on Pattern Recognition (ICPR), Milan, Italy*, 2020
- [26] N.Bonettini, P. Bestagini, S. Milani, S. Tubaro, “On the use of Benford’s law to detect GAN-generated images”, *IAPR International Conference on Pattern Recognition (ICPR), Milan, Italy*, 2020
- [30] S. Mandelli, N. Bonettini, P. Bestagini, S. Tubaro, “Training CNNs in Presence of JPEG Compression: Multimedia Forensics vs Computer Vision”, *International Workshop on Information Forensics and Security (WIFS), New York, USA*, 2020
- [29] N. Bonettini, C.A. Gonano, P. Bestagini, M. Marcon, B. Garavelli, S. Tubaro, “Multitask learning for denoising and analysis of X-ray polymer acquisitions”, *European Signal Processing Conference (EUSIPCO), Dublin, Ireland*, 2021
- [31] S. Mandelli, N. Bonettini, and P. Bestagini, “Source Camera Model Identification”, H. T. Sencar et al. (eds.), *Multimedia Forensics*, Advances in Computer Vision and Pattern Recognition, Springer, 2022
- [32] E.D. Cannas, N. Bonettini, S. Mandelli, P. Bestagini, S. Tubaro, “Amplitude SAR Imagery Splicing Localization”, *IEEE Access*, 2022
- N. Bonettini, C.A. Gonano, P. Bestagini, M. Marcon, B. Garavelli, S. Tubaro, “Comparing AutoEncoder variants for real-time denoising of hyperspectral X-ray”, *IEEE Sensors*, 2022 (submitted)

Part I

Digital Integrity

CHAPTER 2

Sensor Integrity

One of the most interesting problems of image *digital integrity* is assessing the device which took a picture. This problem is important for many reasons, from legal aspects that can be used as evidence in a court to privacy concerns in binding a particular picture to the photographer. There are many methods in the literature aimed to solve this problem. They are usually based on the traces left by the camera sensor on the image. Nevertheless, those methods' reliability and robustness to different post-processing operations are not always studied.

For this reason, in this chapter we investigate: (i) if the device attribution methods can be fooled; (ii) if fooling the device attribution methods leave peculiar traces. In particular, we propose: (i) a method for image anonymization with respect to the camera that shot it (i.e., erasing sensor's traces); (ii) a method for detecting the anonymization of an image.

The chapter is organized as follows: Section 2.1 presents the state of the art of image source attribution. Section 2.2 provides a background on the required concepts we adopt in the chapter. Section 2.3 describes the proposed solution to the image anonymization problem. Section 2.4 describes the proposed solution for exploiting the traces left by anonymization techniques to detect anonymization. Finally, Section 2.5 draws some common conclusions on the proposed methods and concludes the chapter.

2.1 State of the Art

Image source attribution techniques are widely studied in Multimedia Forensics under two different but complementary aspects: camera model identification and camera device identification [33].

Camera model identification aims at finding which camera brand/model shot a specific picture [31]. Techniques that exploit different digital traces left on the captured images have been developed for this task. These solutions can make use of the traces left by Color Filter Array (CFA) interpolation [34, 35], histogram equalization [36], statistical descriptors paired with machine-learning classifiers [37,38], as well as Convolutional Neural Networks (CNNs) [30,39–41].

Camera device identification has proved to be powerful enough to bind a picture to the specific device that shot it. The forensic community [42] has developed a series of multimedia source attribution techniques in the pixel domain that have enabled one to detect which device has been used to acquire an image with precise results [43–45]. These tools do not use any Exchangeable Image File Format (EXIF) information which can easily be modified or removed. This set of tools have been successfully used to establish forensic evidence in major international legal proceedings and have recently passed the United States Judiciary *Daubert* standard [46]. The *Daubert* standard is a rule of evidence regarding the admissibility of expert witnesses' testimony in United States Federal Courts. The FBI Crime Laboratory has thoroughly vetted it and the US National Institute of Justice (NIJ) [47].

The most promising approaches so far exploit the Photo Response Non Uniformity (PRNU) noise [48,49]. A multiplicative noise pattern characterises each imaging sensor, which is inevitably injected into every acquired image. By estimating a noise fingerprint from an image, it is possible to compare it with the PRNU of known camera devices, thus determining which device "took the picture". PRNU-based approaches can also be used on scaled and cropped images [8].

Despite the undeniable need of source attribution methods, 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 threated. Moreover, studying the boundaries of image attribution can enable analysts to be aware of the robustness of camera attribution methods in the presence of malicious attacks. For these reasons, device anonymization techniques tailored to remove or hinder PRNU traces have been developed.

Forensic techniques that focus on removing traces of PRNU from images have been well studied in the literature. We can distinguish two major approaches: the first group requires knowledge of the PRNU pattern to be deleted, whereas the other major group does not require access to the real PRNU to remove it from a given image. Sensor fingerprint removal based on knowledge of the underlying PRNU was first suggested in [43]. This approach assumes a known PRNU fingerprint estimate of a particular imaging sensor is latent in any given image intensity acquired by the same sensor. Hence, the removal of the PRNU fingerprint (the anonymization of the image source) can be achieved by subtraction of the fingerprint estimate from the image intensity. Typical anonymization methods work by blindly modifying pixel values and scrambling their positions in order to make the underlying PRNU unrecognizable. In [10], authors propose to anonymize images by applying seam-carving to change pixel locations, and more recently, [50] compares patch-based methods to shuffle small image blocks. In [9], authors investigate parallel and fast inpainting techniques as methods for image anonymization.

As discussed in the Malicious AI report [51], one should always reflect on the dual-use nature of their work, allowing misuse to influence research priorities and norms.

It is not far-fetched to imagine malevolent agents using the above forensic tools to anonymize images showing sensitive or illegal content. For example, an illegal content producer could anonymize its images to avoid being linked to them in case that its image acquisition equipment was to be seized while in his possession. Hence, it is of paramount importance to know when an image has undergone any process to attempt to remove its underlying PRNU.

2.2 Background

In this section, we report the formal definition of typical PRNU-based forensic problems in order to highlight the goal of our work.

2.2.1 Photo Response Non Uniformity

PRNU is a characterizing feature of the sensor present in digital image acquisition devices that is manifested as a noise term in acquired images [49]. It is due to device-level anomalies in the semiconductors used to manufacture the imaging sensor of digital cameras. Due to the physical origin of these anomalies, PRNU is a unique feature of each camera. It also makes it impractical for the image acquisition pipeline in cameras to effectively compensate for PRNU. Hence, artefacts are present in the digital images produced with the device. PRNU is also robust to lossy compression, which makes it suitable as a robust feature for camera identification [49]. The classic pipeline to estimate the PRNU assumes the availability of a certain number of images coming from the same device to carry out a reliable PRNU estimation. In the basic procedure proposed in the literature [49], PRNU is estimated from a set of N images I_n as:

$$\mathbf{K} = \frac{\sum\limits_{n=1}^{N} \mathbf{W}_n \circ \mathbf{I}_n}{\sum\limits_{n=1}^{N} \mathbf{I}_n^2} \tag{2.1}$$

Being such a powerful forensic footprint, PRNU has been significantly investigated in the literature under various scenarios.

2.2.2 Device attribution

Given an image $\mathbf{I}$ and a generic denoising function $D(\cdot)$, we can compute the noise residual $\mathbf{W} = \mathbf{I} - D(\mathbf{I})$. Given the PRNU fingerprint $\mathbf{K}$ of a camera, we can bind an image $\mathbf{I}$ to that camera if:

$$\mathrm{NCC}(\mathbf{W}, \mathbf{K} \circ \mathbf{I}) > \tau, \tag{2.2}$$

where NCC is the normalized cross-correlation function, $\circ$ denotes the Hadamard (element-wise) product and τ is a threshold set in order to bound false-detection probability below a confidence value α.

2.3 Image anonymization

In this section we describe our method for anonymizing an image to cheat a device identification 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. A Convolutional AutoEncoder (CAE) is trained as anonymization function via back-propagation, exploiting the possibilities offered by a recently introduced CNN-based denoising method [52].

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.

2.3.1 Source device anonymization

Given an image $\mathbf{I}$, an image anonymization function $A(\cdot)$ is a function that generates an anonymized version of $\mathbf{I}$, namely $\hat{\mathbf{I}} = A(\mathbf{I})$. The anonymization process ensures that:

$$\mathrm{NCC}(\hat{\mathbf{W}}, \mathbf{K} \circ \hat{\mathbf{I}}) < \tau, \tag{2.3}$$

where $\hat{\mathbf{W}} = \mathbf{I} - D(\hat{\mathbf{I}})$.

2.3.2 Method

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 $\mathbf{I}$ and its anonymized version $\hat{\mathbf{I}}$; ii) the cross-correlation between the anonymized noise residual $\hat{\mathbf{W}}$ and the camera PRNU $\mathbf{K}$.

Figure 2.1 shows the overall working scheme. An image $\mathbf{I}$ and the corresponding camera PRNU $\mathbf{K}$ are fed as input to the anonymization function $A(\cdot)$. The output of $A(\cdot)$ is an anonymized version of $\mathbf{I}$, namely $\hat{\mathbf{I}}$. The Mean Square Error (MSE) between $\mathbf{I}$ and $\hat{\mathbf{I}}$ is computed and stored into $\mathscr{L}_q$, the first component of the global cost function. The anonymized image $\hat{\mathbf{I}}$ is fed as input to the noise extraction function $N(\cdot)$ and the output $\hat{\mathbf{W}}$ is correlated with the sample-wise product between $\mathbf{K}$ and $\hat{\mathbf{I}}$ to get $\mathscr{L}_c$, the second component of the global cost function. The global cost function $\mathscr{L}$ is then defined as $\mathscr{L} = (1 - \beta) \cdot \mathscr{L}_q + \beta \cdot \mathscr{L}_c$, where β is a weighting parameter tailored at balancing the trade-off between image quality and anonymization performance.

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

To satisfy the gradient computation capability for $N(\cdot)$ we resort to DnCNN [52], a fully-CNN 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.

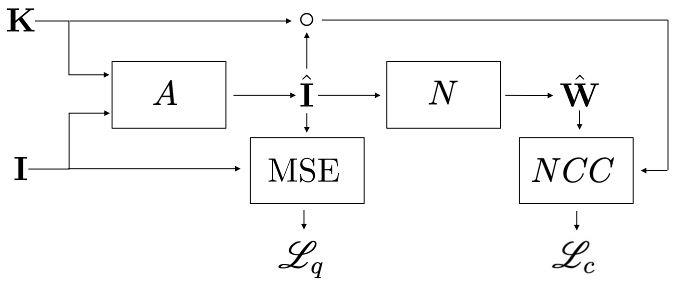


Figure 2.1: *Architecture of the proposed system. An anonymization function* $A(\cdot)$ *is fed with the input image* $\mathbf{I}$ *and the relative camera PRNU* $\mathbf{K}$*. The anonymized image* $\hat{\mathbf{I}}$ *is used to compute a quality loss* $\mathscr{L}_q$ *based on the Mean Squared Error (MSE) between* $\hat{\mathbf{I}}$ *and* $\mathbf{I}$*. The noise residual* $\hat{\mathbf{W}}$*, extracted through a noise extraction function* $N(\cdot)$ *from* $\hat{\mathbf{I}}$*, is used together with the camera PRNU* $\mathbf{K}$ *to determine a correlation loss* $\mathscr{L}_c$*.*

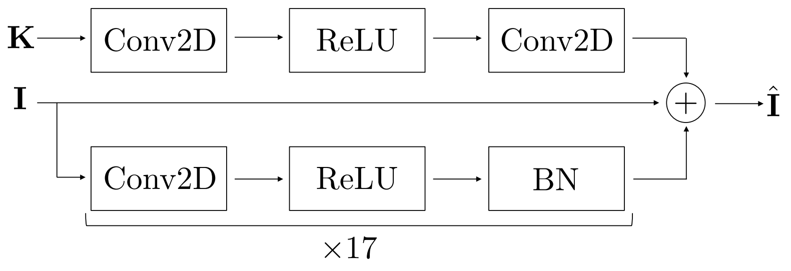


Figure 2.2: *Structure of the proposed CNN-based anonymization function* $A(\cdot)$*. The input image* $\mathbf{I}$ *is processed through a set of* 17 *convolutional layers (Conv2D) followed by ReLU non-linearity and Batch Normalization (BN). The reference PRNU* $\mathbf{K}$ *is processed with two convolutional layers separated by a ReLU non-linearity. The output anonymized image* $\hat{\mathbf{I}}$ *results from the sample-wise algebraic sum of the input image* $\mathbf{I}$ *and the two fully-convolutional branches.*

As for the choice of $A(\cdot)$, we exploit an AutoEncoder (AE) structure similar to DnCNN, as depicted in Figure 2.2. The input image $\mathbf{I}$ is processed by a set of 17 convolutional layers (Conv2D), each followed by ReLU non-linearity and batch normalization (BN). The reference PRNU $\mathbf{K}$ is fed to a convolutional layer, followed by a ReLU and yet another convolutional layer. The final anonymized image $\hat{\mathbf{I}}$ 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 $A(\cdot)$ are learned for every single image via back-propagation, driven by the global cost function $\mathscr{L}$.

The image-wise anonymization process follows as from Algorithm 1. An input image $\mathbf{I}$, a reference PRNU $\mathbf{K}$ and a minimum desired Peak Signal-to-Noise Ratio (PSNR) (PSNR_{min}) are provided as input. The loss weighting factor β is initialized at 0.1. At every iteration the anonymized image $\hat{\mathbf{I}}$ is first computed, together with the MSE loss $\mathscr{L}_q$ and the PSNR P with respect to the original image. Then the noise extraction function $N(\cdot)$ is used to extract a noise residual $\hat{\mathbf{W}}$ from the anonymized image and compute the cross-correlation loss $\mathscr{L}_c$. The global loss $\mathscr{L}$ is computed according to the weighting factor β. As all operations in $A(\cdot)$ are differentiable, it is possible to back-propagate the error and modify $A(\cdot)$ parameters to minimize loss with any iterative optimization algorithm (e.g., Stochastic Gradient Descent (SGD) [53] in our implementation). It is not required for $N(\cdot)$ and $A(\cdot)$ to have a similar structure, as

Algorithm 1 Image-wise anonymization process

```
Require: $\mathbf{I}$, $\mathbf{K}$, $\mathrm{PSNR_{min}}$
  $\beta \leftarrow 0.1$
  for $i$ in $\{1, \dots, 3000\}$ do
    $\hat{\mathbf{I}} \leftarrow A(\mathbf{I}, \mathbf{K})$
    $\mathscr{L}_q \leftarrow \mathrm{MSE}(\mathbf{I}, \hat{\mathbf{I}})$
    $P \leftarrow \mathrm{PSNR}(\mathbf{I}, \hat{\mathbf{I}})$
    $\hat{\mathbf{W}} \leftarrow N(\cdot)(\hat{\mathbf{I}})$
    $\mathscr{L}_c \leftarrow |\ \mathrm{NCC}(\hat{\mathbf{W}}, \mathbf{K} \circ \hat{\mathbf{I}})\ |$
    $\mathscr{L} = (1 - \beta) \cdot \mathscr{L}_q + \beta \cdot \mathscr{L}_c$
    $A(\cdot) \leftarrow \textsc{BackPropagate}(A(\cdot), \mathscr{L})$
    if $\textsc{mod}(i, 300) = 0 \wedge \mathrm{P} < \mathrm{PSNR_{min}}$ then
      $\beta \leftarrow \beta/4$
    end if
    if $\mathrm{P} > \mathrm{PSNR_{min}} \wedge \mathscr{L}_c < 10^{-4}$ then
      return $\hat{\mathbf{I}}$
    end if
  end for
  return $\hat{\mathbf{I}}$
```

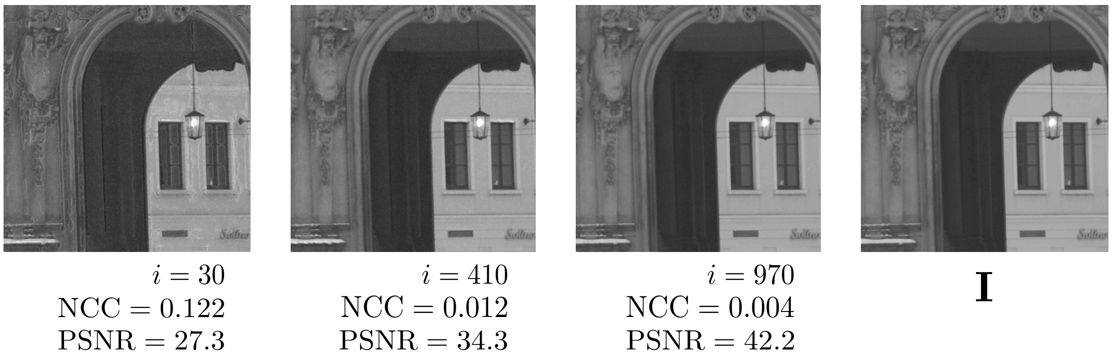


Figure 2.3: *Iterations of the proposed algorithm on a sample image. From left to right the evolution of* $\hat{\mathbf{I}}$ *at* $i = \{30, 310, 970\}$ *with cross-correlation* NCC *decreasing and PSNR* P *increasing. The rightmost picture is the original image* $\mathbf{I}$.

long as both are differentiable operators. Once every 300 iterations if the PSNR value P is smaller than the desired minimum value $\mathrm{PSNR_{min}}$ the weighting factor β is reduced by a factor 4, to raise the importance of the MSE loss $\mathscr{L}_q$ over the cross-correlation loss $\mathscr{L}_c$.

If the current PSNR is greater than the desired minimum and the cross-correlation loss is small enough (i.e., $\mathscr{L}_c < 10^{-4}$), the current anonymized image $\hat{\mathbf{I}}$ 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 $\hat{\mathbf{I}}$, NCC and P over the iteration is provided in Figure 2.3.

2.3.3 Experimental Setup

To state the effectiveness of the proposed approach, we resort to the same dataset and metrics used in [9, 50]. 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 [54]. All the images are cropped in their center to a fixed size of 512×512 pixels. 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 $N_{\text{dn}}(\cdot)$; ii) the Wavelet-based noise exaction function [55] commonly used in PRNU-based works, denoted as $N_{\text{wl}}(\cdot)$. As for the use of DnCNN as noise extractor, we resort to the pre-trained model available from [52]. We resort to Pytorch [56] as Deep Learning and CNN framework.

The reference PRNU $\mathbf{K}$ for each device is estimated from 25 raw flatfield images from the same database, according to $N_{\text{wl}}(\cdot)$ as from [43]. All the 600 images are anonymized by varying the PSNR$_{\text{min}}$ 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$_{\text{min}}$ we observe the distribution of the obtained PSNR values. Noise residuals are extracted with $N_{\text{dn}}(\cdot)$ and $N_{\text{wl}}(\cdot)$ for each anonymized image and than correlated with the 6 camera PRNUs. For each PSNR$_{\text{min}}$ 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 (TPR) value at a False Positive Rate (FPR) $\alpha = 0.01$ (TPR@0.01) and the Area Under the Curve (AUC). Small AUC values indicate good anonymization performance. Small TPR@0.01 values indicate that when accepting a small FPR it is not possible to bind the picture to its camera device.

2.3.4 Results

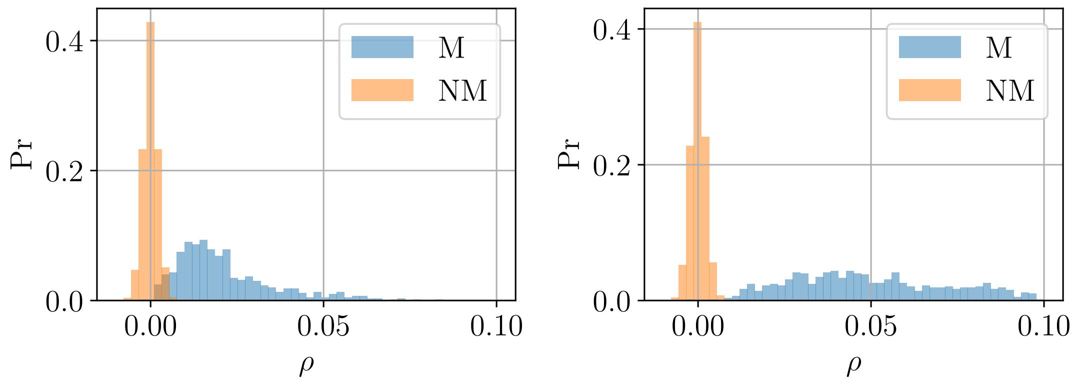
In this section we report a set of results to test and validate the proposed pipeline. Before describing the anonymization results, we do some considerations on the denoising operator and the PSNR.

Validation of Denoising Operator

First, we need to asses whether DnCNN ($N_{\text{dn}}(\cdot)$) can be used as a reasonable approximation for the widespread Wavelet ($N_{\text{wl}}(\cdot)$) noise extractor tailored to PRNU matching and camera device identification. Figure 2.4 shows the distribution of NCC when $N_{\text{dn}}(\cdot)$ and $N_{\text{wl}}(\cdot)$ are used as noise extractors from pristine images. In both cases the reference PRNU ($\mathbf{K}$) 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 2.5 shows the difference in terms of ROC between $N_{\text{dn}}(\cdot)$ and $N_{\text{wl}}(\cdot)$ 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$_{\text{min}}$ as a driving criteria for the minimum accepted image quality, we are interested in checking whether this criteria is actually met in an experimental fashion. In facts, it might happen that the anonymization loop reaches the maximum number of iterations but the PSNR between $\mathbf{I}$ and $\check{\mathbf{I}}$ is still smaller



(a) $N_{dn}\left(\cdot\right)$ **(b)** $N_{wl}\left(\cdot\right)$

Figure 2.4: *Distribution of normalized cross-correlation values on pristine images using DnCNN (a) and Wavelet (b) noise extractors, for matching image-PRNU pairs (M) and non-matching pairs (NM).*

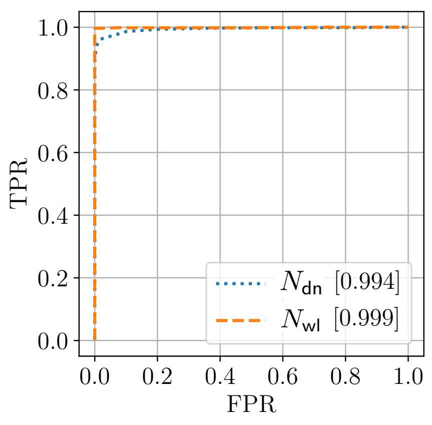


Figure 2.5: *Comparison between $N_{wl}\left(\cdot\right)$ and $N_{dn}\left(\cdot\right)$ as noise residual extractors in terms of ROC. The AUC reported between squared brackets shows almost equivalent performance in terms of detection.*

than $\mathrm{PSNR}_{\mathrm{min}}$. Figure 2.6 reports the histograms of PSNR values obtained for various values of $\mathrm{PSNR}_{\mathrm{min}}$. It is possible to notice that for every choice of $\mathrm{PSNR}_{\mathrm{min}}$ 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 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 NCC values obtained from anonymized images with noise residuals extracted with DnCNN ($N_{\mathrm{dn}}\left(\cdot\right)$) and Wavelet ($N_{\mathrm{wl}}\left(\cdot\right)$). Figure 2.7a shows how the distributions of matching and non-matching NCC values, obtained when noise residuals are extracted from $\hat{\mathbf{I}}$ through $N_{\mathrm{dn}}\left(\cdot\right)$, 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-

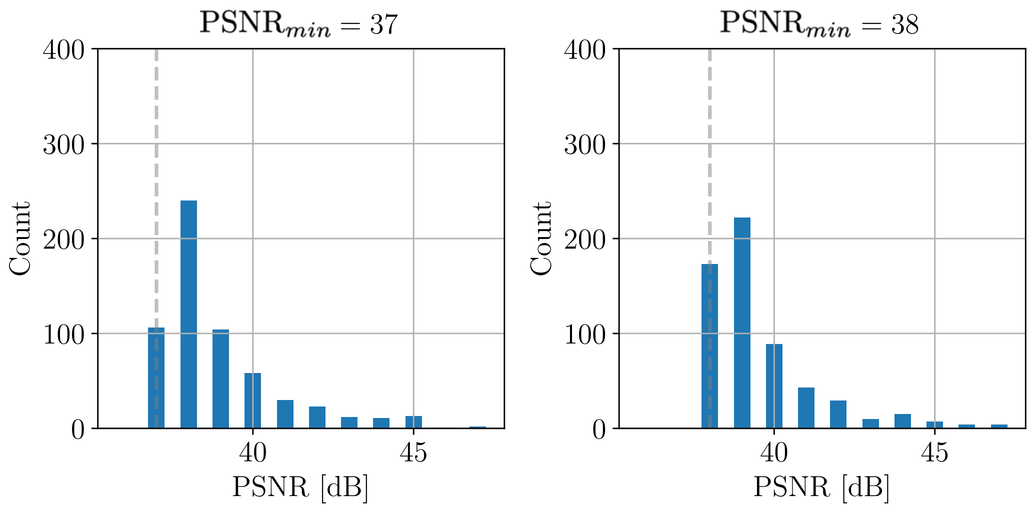
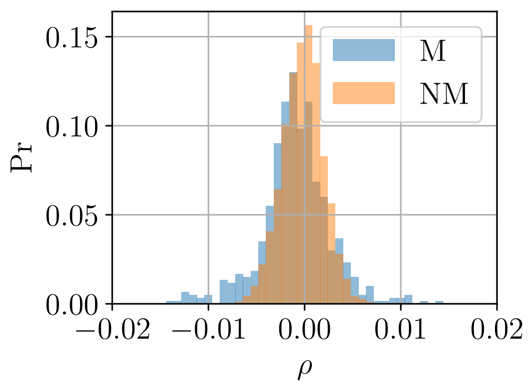


Figure 2.6: *Real PSNR distribution when varying PSNR$_{min}$ in $\{37, 38\}$. The real PSNR values are always greater or equal the the minimum value (vertical dashed gray line). The same behavior is obtained for different PSNR$_{min}$ values.*

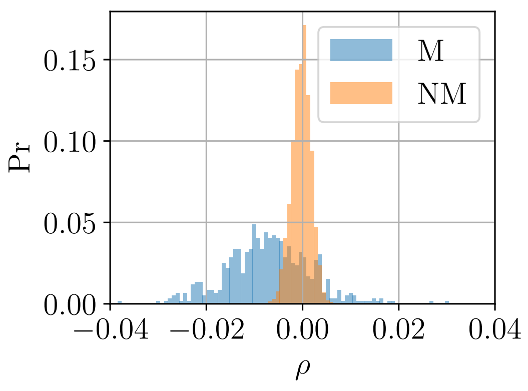
correlation between the reference PRNU $\mathbf{K}$ and the noise residual extracted through $N_{\text{dn}}(\cdot)$. As we wish to evaluate the effect of the proposed method when the Wavelet-based noise extractor is used on $\hat{\mathbf{I}}$, Figure 2.7b shows the distribution of matching and non-matching NCC values obtained when noise residuals are extracted with $N_{\text{wl}}(\cdot)$. We can immediately spot two differences with respect to the $N_{\text{dn}}(\cdot)$ 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 defined in Equation (2.2) to asses whether an image $\hat{\mathbf{I}}$ under investigation comes from a camera whose PRNU is $\mathbf{K}$. However, a smart investigator would also perform another test, evaluating the absolute value of the normalized cross-correlation, thus building a symmetric test $|\text{NCC}(\mathbf{W}, \mathbf{K} \circ \mathbf{I})| > \tau$. In the plots, we refer to the results obtained with the standard Wavelet detector with $N_{\text{wl}}(\cdot)$, while the results obtained with the Wavelet symmetric detector are denoted as $N^{\text{a}}_{\text{wl}}(\cdot)$.

Figure 2.8 shows the ROC on anonymized images detection for PSNR$_{\text{min}} = 40$. If $N_{\text{dn}}(\cdot)$ is used to extract the noise residual from $\hat{\mathbf{I}}$ we get almost perfect anonymization performance. This confirms that the anonymization loop, based on the minimization of the cross-correlation value between $\mathbf{K} \circ \hat{\mathbf{I}}$ and $\hat{\mathbf{W}}$ extracted through $N_{\text{dn}}(\cdot)$, is effectively working as expected. When noise residuals are extracted from $\hat{\mathbf{I}}$ through the Wavelet-based function and the unidirectional test in Equation (2.2) is used ($N_{\text{wl}}(\cdot)$), the detection performance are severely affected. However, resorting to the symmetric detector ($N^{\text{a}}_{\text{wl}}(\cdot)$) 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 2.9, where two standard metrics in anonymization are presented. Figure 2.9a and Figure 2.9b respectively report the TPR at a fixed FPR of 0.01 and the AUC for several median PSNR values. Each point is obtained by set-



(a) $N_{dn}(\cdot)$



(b) $N_{wl}(\cdot)$

Figure 2.7: *Distribution of normalized cross-correlation values on anonymized images using DnCNN (a) and Wavelet (b) noise extractors, for matching image-PRNU pairs (M) and non-matching pairs (NM).*

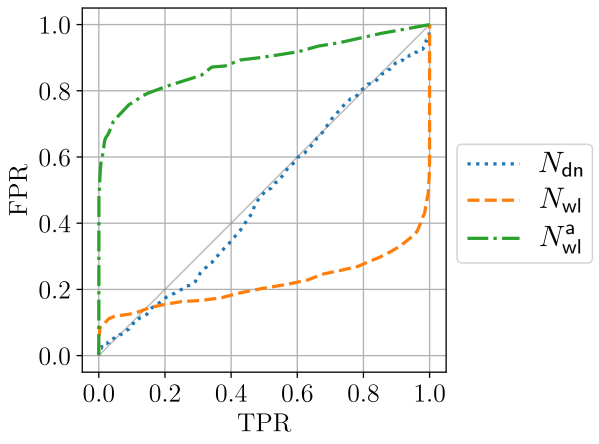
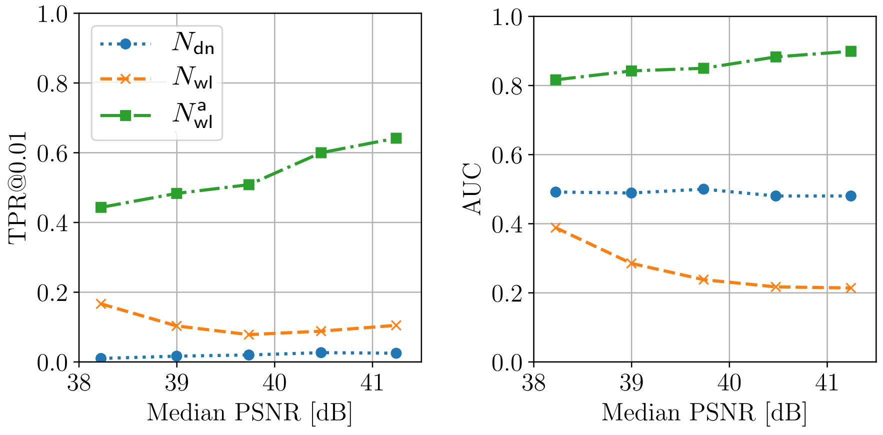


Figure 2.8: *ROC for PSNR$_{min}$* $= 40$.

ting PSNR$_{\text{min}}$ to $\{37, 38, 39, 40, 41\}$. The almost zero TPR@0.01 value for $N_{\text{dn}}(\cdot)$ 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 ($N^{\text{a}}_{\text{wl}}(\cdot)$) the anonymization is not guaranteed anymore.

2.4 Anonymization detection

In this section, we propose a method for detecting the anonymization of an image. The method we propose is based on the use of a CNN. However, instead of feeding the network with images in the pixel domain, we show that applying the proposed pre-processing technique dramatically improves the performance of anonymization detection. The proposed solution can generalise to different kinds of anonymization methods never seen during training, thus showing that using specific domain knowledge can help



(a) (b)

Figure 2.9: *TPR at a fixed FPR* $\alpha = 0.01$ *(a) and AUC (b) when varying PSNR*$_{min}$.

pure data-driven techniques.

2.4.1 Problem statement

We define anonymization detection problem as a two-class classification problem, where $\mathcal{C}_0$ is the class of original images and $\mathcal{C}_1$ is the class of the anonymized images.

Given an image $\mathbf{I}$ under investigation, an anonymization detector is an operator $M(\cdot)$ such that:

$$\hat{y} = M(\mathbf{I}), \tag{2.4}$$

where $\hat{y} \in \{0, 1\}$ represents a label that assigns $\mathbf{I}$ to $\mathcal{C}_0$ or $\mathcal{C}_1$. Our goal is to design an operator $M(\cdot)$.

2.4.2 Method

Given an image under analysis $\mathbf{I}$, our goal is to detect whether its PRNU traces have been removed or not. We propose the following method to determine this: (i) we pre-process the image in order to extract a feature that exposes salient anonymization information; (ii) we add the proposed feature to a CNN that identifies whether the analyzed image has been anonymized or not.

Feature extraction

Despite the well-known capabilities of CNNs to work directly in the pixel domain, they can yield better performance when coupled with domain specific knowledge of the problem to be tackled. Using domain knowledge as indicated in [57, 58], we leverage the efforts of the forensics community and propose a preprocessing approach to extract the residual noise left in the input image and then shift it into the Fourier domain. Due to the noisy nature of the PRNU pattern and the subtlety of the traces left by the anonymization techniques, if pixel domain information were to be input into the neural network, this would lead to poorly trained models incapable of generalization or overfitted on image features instead of PRNU removal detection. However, our

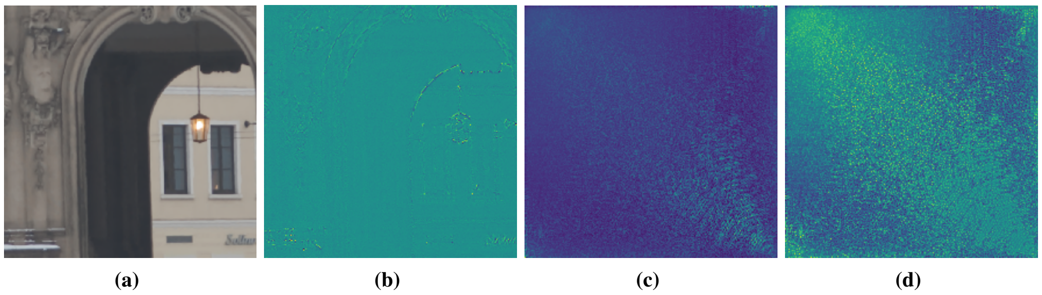


Figure 2.10: *(a) RGB image anonymized using [50]. (b) Wavelet noise extracted from the image. (c)* DFT_2 *of the signal. (d) Final feature after Wiener filtering.*

proposed preprocessing leads to a boost in the detection accuracy of our trained neural network model while generalizing to unseen camera models, as shown in the Results section.

Our feature extraction method is defined as follows. Let us consider a grayscale $H \times W$ image $\mathbf{I}$. By means of the Wavelet denoising function $D_{\mathrm{w}}(\cdot)$ proposed in [55] and often used for PRNU estimation, we compute the noise residual $\mathbf{W}$ from the image as:

$$\mathbf{W} = \mathbf{I} - D_{\mathrm{w}}(\mathbf{I}). \tag{2.5}$$

We then compute the magnitude of the 2D Discrete Fourier Transform (DFT_2) as:

$$\mathbf{W}_{\mathrm{F}} = |\mathrm{DFT}_2(\mathbf{W})|, \tag{2.6}$$

and we then Wiener filter this signal, following the method described in [43]:

$$[\mathbf{F}]_{ij} = [\mathbf{W}_{\mathrm{F}}]_{ij} \cdot \frac{\sigma_s^2}{[\mathbf{S}_{\mathrm{W}}]_{ij} + \sigma_s^2}, \tag{2.7}$$

with:

$$\sigma_s^2 = \delta \cdot \sigma_{\mathrm{W}}^2, \tag{2.8}$$

where $i = 1, \dots, H$, $j = 1, \dots, W$, σ_{W}^2 is the variance of $\mathbf{W}_{\mathrm{F}}$ and $\mathbf{S}_{\mathrm{W}}$ is the matrix containing the variance of the energy of $\mathbf{W}_{\mathrm{F}}$ computed over a 3×3 moving window. The parameter δ must be chosen depending on the input signals in order to drive the Wiener filtering operation (we set $\delta = 0.77$ in our experiments). Without loss of generality, we iterate this procedure over each of the three channels of a RGB image. The result is a $H \times W \times 3$ feature $\mathbf{F} = F(\mathbf{I})$ we use as network input. Figure 2.10 visually displays all the feature extraction steps.

Model

Due to the formulation of our problem, we can make use of transfer learning, which is known to increase the performance of detection models in limited data settings [59]. This means that we can use CNN models pretrained on large image datasets and leverage their learnt filters. After selecting a suitable architecture, we use transfer learning on a model trained on ImageNet by replacing its last fully-connected layer with a $(n_l, 1)$ fully-connected layer, where n_l is the number of input features of the original last layer. Additionally, we perform a sigmoid over the network output to bind it to $[0, 1]$. Thus, giving as input a batch of B samples $\mathbf{X}$ of size $B \times H \times W \times 3$ and its label tensor $\mathbf{y}$ of size $B \times 1 \in \{0, 1\}$, the network output $\hat{\mathbf{y}}$ of size $B \times 1 \in [0, 1]$ is a tensor of scalars representing the likelihood of each sample to be anonymized. We choose Binary Cross Entropy as a loss function between the target and the output:

$$\mathscr{L}_b = -\left[y_b \cdot \log \hat{y}_b + (1 - y_b) \cdot \log\left(1 - \hat{y}_b\right)\right], \tag{2.9}$$

where $b \in \{1, ..., B\}$ is the sample in-batch index. We train each model using Adam optimizer [60] until reaching a validation plateau.

In our experiment, we consider ResNet [61] as CNN model. AlexNet [62] and VGG [63] were also considered, but underperformed ResNet, probably due to the consistently higher number of parameters.

2.4.3 Experiments

In this section we describe the dataset, the training strategy, and the different kinds of analysis we performed to highlight different aspects of the proposed solution.

Dataset

Starting from the 600 original images from the Dresden database [54] used in [9], we applied to each image the two anonymization procedures described in [9] and [64], thus obtaining a corpus of $600 \times 3 = 1800$ images (i.e., the original images, and the two sets of anonymzed ones). To the best of our knowledge, these are the most recent methods for blind device anonymization known in literature. In order to validate the proposed approach on various kinds of anonymized images, as well as to test cross-dataset generalization, starting from the above images we define different sets of data. Specifically, we define three training dataset:

- $\mathcal{D}_1$ composed by 300 original images and 300 images anonymized following [9],
- $\mathcal{D}_2$ composed by the same 300 original images and 300 images anonymized following [64],
- $\mathcal{D}_3$ composed by the same 300 original images, the first 150 images anonymized by [9] and the last 150 images anonymized by [64].

Following the same strategy, we construct two testing sets with the remaining images:

- $\mathcal{T}_1$ composed by 300 original images and 300 images anonymized following [9],
- $\mathcal{T}_2$ composed by the same 300 original images and 300 images anonymized following [64].

All the images are RGB images and have been central-cropped to size 512×512 pixels. To increase the number of available samples, we split each image in 224×224 blocks with an overlapping stride of 32×32. During training, we use 70% of each training set for training, and the remaining 30% for validation. In total, we have 42000 samples in each training set, 18000 in each validation set and 60000 in each test set.

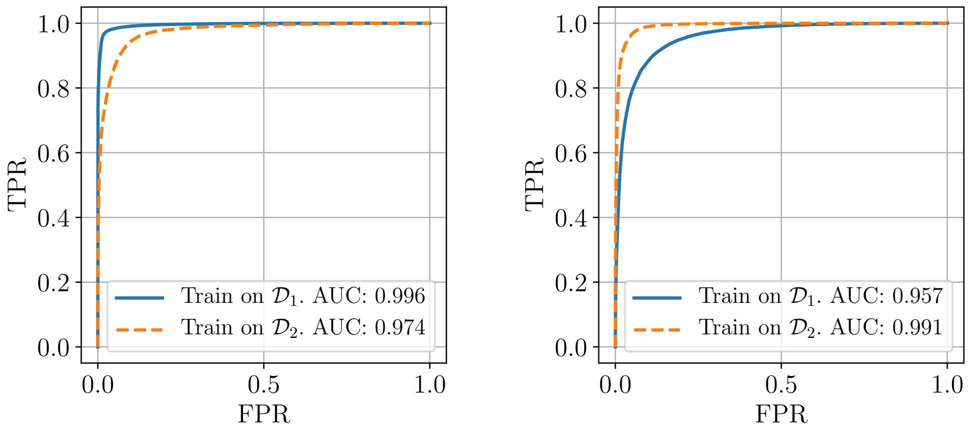


Figure 2.11: *ROC curves for two different testing dataset. Left: Test on $\mathcal{T}_1$; right: Test on $\mathcal{T}_2$.*

2.4.4 Results

In this section we provide the results of the conducted experiments. We evaluate the cross dataset scenario, the Leave One Out scenario, and we perform some additional experiments on the robustness to transformations.

Cross dataset results

During the test phase we examined the models trained on $\mathcal{D}_1$, $\mathcal{D}_2$, $\mathcal{D}_3$ and we test them against $\mathcal{T}_1$ and $\mathcal{T}_2$.

We perform our test in very challenging scenarios, including training on a specific anonymization method and testing on the other one. The test procedure consists in freezing the network weights and predicting $\hat{y}$ given as input the feature $\mathbf{F}$. By thresholding $\hat{y}$ with different thresholds, we compute TPR and FPR w.r.t. the true label y and plot them as a ROC curve. We show in Figure 2.11 the ROC curves for the most challenging training and testing conditions. The AUC on same-type dataset are very high and we are able to reach comparable AUC even in the worst case scenario in which we train on $\mathcal{D}_1$ and test on $\mathcal{T}_2$ and viceversa. This shows that the proposed method generalizes over different anonymization methods, and does not simply overfit to recognize one.

Figure 2.12 shows the network performances with different input preprocessing (image $\mathbf{I}$ or the noise residual $\mathbf{W}$ or the feature $\mathbf{F}$). The designed feature $\mathbf{F}$ is the one that shows the best results regardless the arduous testing scenarios.

Leave One Out

One might be concerned the proposed CNN model learns to recognize just the PRNU of the devices used during training.

To verify the model performance with unseen camera devices, we design a Leave One Out testing procedure. We modify our training dataset $\mathcal{D}_3$ by removing all the images acquired with device `Nikon_D200_0`, creating $\tilde{\mathcal{D}}_3$. Similarly, we remove all the images from all the devices except `Nikon_D200_0` from $\mathcal{T}_1$ and $\mathcal{T}_2$ and we add to them the images removed from $\mathcal{D}_3$. These two new mono-device datasets are known

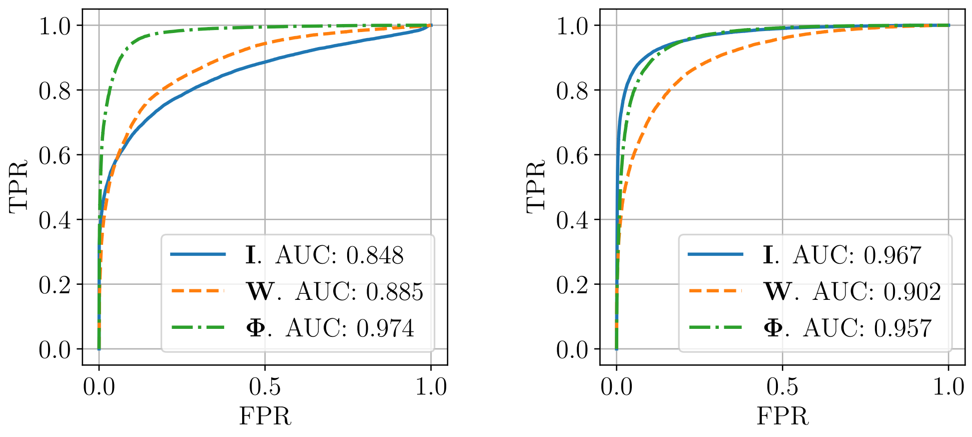


Figure 2.12: *ROC curves with various preprocessing approaches. Left: Train on $\mathcal{D}_2$, test on $\mathcal{T}_1$; right: Train on $\mathcal{D}_1$, test on $\mathcal{T}_2$.*

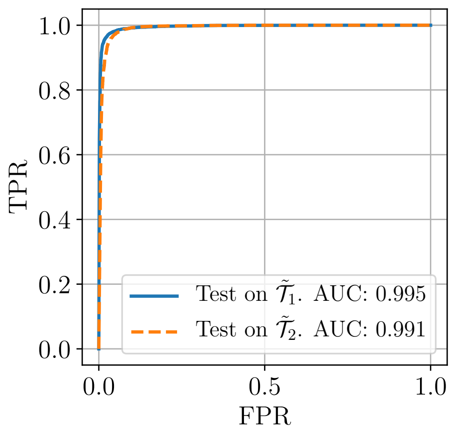


Figure 2.13: *ROC for Leave One Out strategy. Train on $\tilde{\mathcal{D}}_3$ test on $\tilde{\mathcal{T}}_1$and $\tilde{\mathcal{T}}_2$.*

as $\tilde{\mathcal{T}}_1$ and $\tilde{\mathcal{T}}_2$. Figure 2.13 shows the ROC for training on $\tilde{\mathcal{D}}_3$ and testing on $\tilde{\mathcal{T}}_1$ and $\tilde{\mathcal{T}}_2$. We can assess that our model is capable of discriminating between original and anonymized images even when it is tested against a device which was not present in the training set.

Robustness to transformations

As it is clear from the literature [65], PRNU can be somehow corrupted by other editing operations. We are therefore interested in studying whether our method recognizes these situations or not, and we design a proper testing strategy. In the test phase, before extracting the feature matrix **F** from the image **I**, we compute:

$$\mathrm{NCC}_{pre} = \mathrm{NCC}(\mathbf{W}, \mathbf{IK}), \tag{2.10}$$

where $\mathbf{W}$ is the noise residual obtained from **I** and **K** is the reference PRNU. NCC_{pre} gives us a baseline metric of correlation between an image and its noise fingerprint.

Then we modify the image with one of the available transformations in Table 2.1,

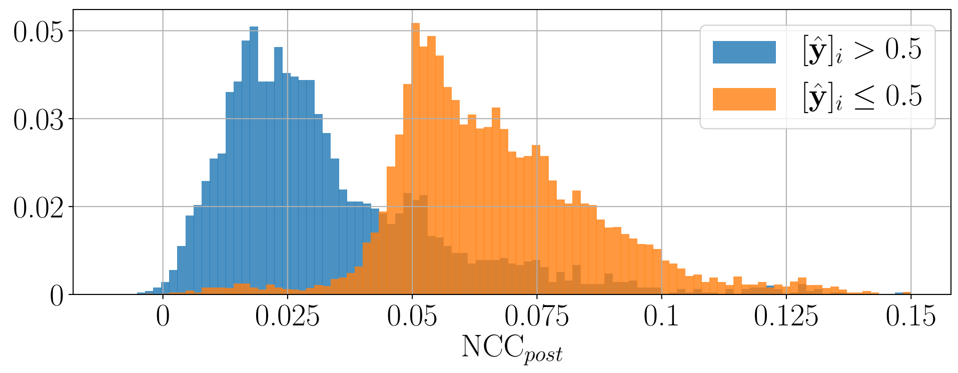


Figure 2.14: *Distribution of* NCC_{post} *values.*

Table 2.1: *Transformation table with their parameters set. Parameter for JPEG is the Quality Factor, for Gamma is the exponent, for Brightness ranges from 0 (black image) to 1 (original image), for Contrast ranges from 0 (solid gray image) to 1 (original image).*

Transformation	Parameters set
JPEG compression	$70, 75, 80, 85, 90$
Gamma correction	$0.5, 0.6, 0.7, 0.8, 0.9$
Brightness correction	$0.5, 0.6, 0.7, 0.8, 0.9$
Contrast correction	$0.5, 0.6, 0.7, 0.8, 0.9$

randomly selecting it and a parameter from its set, and we compute:

$$\text{NCC}_{post} = \text{NCC}(\mathbf{W}_t, \mathbf{I}_t\mathbf{K}), \tag{2.11}$$

where $\mathbf{I}_t$ and $\mathbf{W}_t$ denote the transformed image and the noise residual obtained from it, respectively. NCC_{post} gives us a measure of the degradation of the fingerprint introduced by the transformation. It is worth noting that all the transformations in Table 2.1 are non-geometric transformations, hence we do not need to transform the reference PRNU $\mathbf{K}$ too. After computing these two metrics, we compute the network output $\hat{\mathbf{y}}$ from $\mathbf{F}$. Figure 2.14 shows the distribution of NCC_{post} values. For the sake of visualization, we show only the samples with $\text{NCC}_{pre} > 0.05$, i.e., we select only the images that matches with the PRNU *before* the transformation. By observing the distribution of NCC_{post} we notice that some images have values < 0.05, meaning that some transformations degrade the PRNU traces to the point that the correlation test fails. After computing the network scores $\hat{\mathbf{y}}$, we can clearly distinguish two value distributions in the plot, the left one with $[\hat{\mathbf{y}}]_i > 0.5$ and the right one with $[\hat{\mathbf{y}}]_i \leq 0.5$. This shows that our method is robust to transformations: when the NCC value is low the network classifies the sample as anonymized, whereas when the NCC value is high, the network classifies the sample as original. More in general this result highlights the fact that we are not simply learning to detect the artefacts left by the two considered anonymization methods, but we are able to effectively discriminate between correlating and non-correlating images.

2.5 Conclusions

In this chapter we considered the problem of image anonymization and its detection. We 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. We believe our 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 used 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. Regarding the detection, we proposed a new kind of feature starting from RGB images, and use this feature as input to a CNN. We select two different blind anonymization techniques and perform network finetuning on images processed with these techniques. Results show the effectiveness of our method comparing to several different preprocessing pipelines. Our model is capable of generalizing over unseen devices and it is robust against non-geometric transformations. Despite the undoubted capability of CNNs, pure data-driven approach was not sufficient for solving the problem well enough. Forensics domain knowledge allowed us to carefully design a preprocessing pipeline for feature extraction to ease network training, thus showing that model-based and data-driven methods can benefit from one another.

CHAPTER 3

Coding Integrity

The vast majority of digital images are stored in a compressed format. Compression is often done directly by the device that acquires the image, and almost every online platform (e.g., social media, chat applications) applies some compression to the uploaded images. For this reason, a possible way to verify the *digital integrity* of pictures is to study the traces left by the codec algorithms. The JPEG format is the most used compression format over the internet, as it dramatically reduces the file size while retaining visual quality.

For this reason, in this chapter, we treat the *digital integrity* of images that have undergone JPEG compression. Indeed, it is of great interest to look for inconsistencies or multiple coding steps that could reveal tampering with the image. We first propose a data-driven method for detecting double JPEG compression. Then, we extend the analysis to multiple JPEG compression, up to four coding steps. Finally, we propose a method to spot inconsistencies into multiple compressions, in the case they are performed with different implementations of the JPEG algorithm.

Section 3.1 provides the state of the art of JPEG forensics. Section 3.2 introduces background concepts on the JPEG compression algorithm. Section 3.3 describes the proposed method for detecting double JPEG compression in the two possible scenarios of aligned and non-aligned second compression. Section 3.4 describes the proposed method for detecting up to four JPEG coding steps. Section 3.5 describes the proposed method for detecting double compression with a different implementation of the JPEG algorithm. Finally, Section 3.6 draws a common conclusion on the three proposed methods, ending the chapter.

3.1 State of the Art

Due to the wide availability of easy-to-use imaging software in the last decades, the diffusion of tampered content has become a widespread phenomenon. Among the techniques developed by the image forensic community to fight this trend [66, 67], great attention has been devoted to methods analyzing JPEG traces [11, 68]. 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 significant attention in image forensics, and the presence of tampering is often revealed by looking for the artefacts left by JPEG re-compression. However, different artefacts are introduced depending on whether the second JPEG compression grid is aligned or not with the one adopted by the first compression. For this reason, these two scenarios are often analyzed separately. They are commonly referred to Aligned Double JPEG (A-DJPEG) compression detection and Non Aligned Double JPEG (NA-DJPEG) compression detection.

In many cases, manipulation takes place only on limited parts of the image. Therefore Double JPEG (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 the localization of manipulated regions in image forgery detection problems. However, most of the techniques performing double JPEG detection in the literature focus on estimating the 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.

It is well known that double JPEG compression leaves peculiar artefacts in the Discrete Cosine Transform (DCT) domain, in particular, on histograms of block-DCT coefficients [68]. Accordingly, many proposed detection algorithms focus on analysing first-order statistics of DCT coefficients. This is the case with the data-driven approach in [69], based on analysis of low-frequency block-DCT coefficients histograms, and many model-based approaches, e.g., the ones in [70–72] that rely on the distribution of First Digits (FDs) in block-DCT coefficients and methods based on Benford-Fourier analysis [14, 73]. Data-driven detectors based on features derived from second-order statistics have also been proposed, e.g., [11]. 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. Moreover, 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, [74] exploits the double quantization 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 [75] through the use of an improved probability model. However, spatial resolution considered by the authors for good detection accuracy with these methods is 256×256, and performances drop significantly when smaller regions are considered. Besides, this method performs poorly when the Quality Factor (QF) used for the first compression (i.e., QF1) is significantly larger than the second one (i.e., QF2). In [76], localization of spliced regions is achieved by using FSD features of block-DCT coefficients and employing a Support Vector Machine (SVM) classifier. More recently, in [13], authors proposed a novel method that relies on a one-dimensional Convolutional Neural Network (CNN), designed to learn discriminant features from DCT coefficients histograms automatically. This approach outperforms both methods in [75] and [76], achieving good detection performances with small sized images up to 64×64 pixel. However, all the above approaches exploit the peculiar traces left by A-DJPEG compression and 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 domains [77, 78] and DCT domain [12, 79]. Specifically, in [78] authors proposed a method able to detect both aligned and non-aligned re-compression. The scheme works by combining periodic artefacts in spatial and frequency domains. Specifically, a set of features is computed to measure the periodicity of blocking artefacts, which is altered when an NA-DJPEG compression occurs. Another set of features is used to measure the periodicity of DCT coefficients, which is perturbed in presence of A-DJPEG. This approach for non-aligned re-compression detection is outperformed by [12]. Furthermore, in [80], authors 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 [75], where a unified statistical model characterizing JPEG artefacts in the DCT domain is considered. However, similarly to [75] (and [12]), 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.

Recently, research on image forensics has started also to analyze chains of processing operations in order to model more realistic scenarios [81]. Nevertheless, the issue of identifying multiple JPEG compressions is, by far, a less investigated problem. Indeed, in many practical situations, pictures under analysis might be compressed several times. Think for example to the increasingly widespread habit of sharing visual content on social media: the average user typically shot a picture with a smartphone (first compression), share it through a messaging app (second compression), and the receiver may re-share it or post it on a social platform (third compression). As a consequence, this operation necessarily results in an “at-least-three-compression” chain for the image under analysis.

Given the strong likelihood of digital images to undergo more than two compression stages, finding a method able to estimate the number of endured JPEG compressions is of paramount importance for the reconstruction of processing history of the investigated content. In this vein, the method proposed in [82] aims at identifying up to three JPEG compressions through a testing scheme based on the statistical analysis of Benford-Fourier coefficients [83]. The problem of detecting up to four JPEG compressions is addressed in [84] by exploiting the FD of DCT coefficients in absolute values. The algorithm is based on SVMs and allows to estimate up to four compression cycles.

Another interesting scenario to consider is the detection of multiple compressions in case they are done by means of different software. The authors of [85] have recently shown that it is possible to detect whether an image has been compressed with different JPEG implementations according to the used quantization rule (i.e., flooring, ceiling or rounding).

3.2 Background

In this section we provide a fast overview on JPEG compression algorithm to highlight some of the founding concepts needed to understand the rest of the work.

3.2.1 JPEG compression

JPEG is a lossy image transform coding technique based on block-wise 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 predefined quantization tables scaled by a QF. A lower QF indicates a stronger quantization, thus a lower quality of the final decompressed image.

Double compression occurs when an image compressed with a $\text{QF} = \text{QF1}$ is first decompressed and then compressed again with $\text{QF} = \text{QF2}$. If no operations are applied between the two compression steps, the 8×8 pixel blocks left by the first and second JPEG 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.

In decoding phase, binary stream is decompressed, coded blocks are reconstructed by applying inverse DCT on rescaled coefficients and the image is re-built in the pixel domain [86].

Due to quantization, it is well-known in the literature that histograms of DCT coefficients show a typical comb-like shape, and spacing between consecutive peaks is related to the adopted quantization step size. Moreover, if an image is encoded many times with different QFs, the resulting quantization levels are modified accordingly [68].

3.3 Double JPEG Compression Detection based on Convolutional Neural Networks

In this section we detail the proposed CNN-based solutions for aligned and non-aligned double JPEG compression detection.

3.3.1 Problem Formulation

Our goal is to build a detector which is able to classify between single compressed and double compressed images. In other words, let $\mathcal{H}_0$ correspond to the hypothesis of single compressed image, and $\mathcal{H}_1$ to the hypothesis of image compressed twice. Given a $L \times B$ pixel image $\mathbf{I}$, we want to detect whether $\mathcal{H}_0$ or $\mathcal{H}_1$ 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. As it will be further explained in Section 3.3.4, the investigation of different approaches is motivated by the fact that aligned and non-aligned

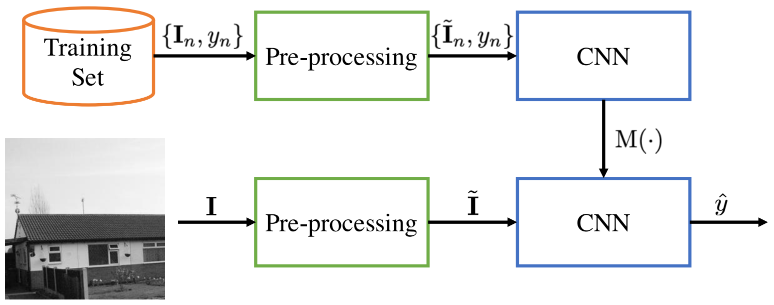


Figure 3.1: *Pipeline common to the proposed solutions. CNN training (top) is performed using images* $\mathbf{I}_n$ *labeled with* y_n*. The CNN model* $M(\cdot)$ *is then used for testing (bottom) a new image* $\mathbf{I}$ *and obtain the candidate label* $\hat{y}$*. Optional pre-processing might be applied to the images.*

DJPEG compressions leave different footprints and then in principle cannot be detected in the same way.

3.3.2 Proposed Solutions

The proposed methodologies follow a common pipeline depicted in Figure 3.1 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 $\{\mathbf{I}_n, y_n\}$, $n \in [1, N]$, where $y_n = 0$ if image y_n verifies $\mathcal{H}_0$ (single compressed), $y_n = 1$ if it verifies $\mathcal{H}_1$ (double compressed). After training, the CNN becomes the learned model $M(\cdot)$ 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 $\mathbf{I}_n$ into $\tilde{\mathbf{I}}_n$. When an image $\mathbf{I}$ is under analysis, it is fed to the trained CNN. The network outputs the probability of the image to verify whether $\mathcal{H}_0$ is true or not. This probability (soft output) is converted to the estimated label $\hat{y}$ by thresholding (hard output). Clearly, if pre-processing is applied during training, it must be applied also during testing.

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

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 [87, 88].

In this case, $\mathbf{I}_n$ corresponds to the JPEG images in the pixel domain (decompressed image) and[1]:

$$\tilde{\mathbf{I}}_n = \mathbf{I}_n - \frac{1}{N}\sum_{n=1}^{N}\mathbf{I}_n. \tag{3.1}$$

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

[1]The operation is performed pixel-wise.

Table 3.1: *Reference CNN architecture parameters. Input-output relations for each layer are reported as function of the input image size* $L \times L \times 1$.

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

Regarding the CNN architecture, we resort to a slightly deeper variation of the well-known LeNet [89] developed for digits recognition, which has already been successfully exploited for forensic analysis [40, 90, 91]. This network architecture is depicted in Figure 3.2 (bottom part) and input-output size of each layer are reported in Table 3.1. $L \times L$ 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 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.

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 $\tilde{\mathbf{I}}_n$ corresponds to the noise residual

$$\tilde{\mathbf{I}}_n = \mathbf{I}_n - F\left(\mathbf{I}_n\right), \tag{3.2}$$

where $F\left(\cdot\right)$ is the denoising operator described in [55], 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 (MSE) estimation. This algorithm is widely used in forensics for its good capability of separating image content from noise [43]. With regard to the CNN architecture, we rely again on the one described in Table 3.1.

CNN Embedding DCT Histograms

The above solutions implicitly assume that DJPEG artefacts 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 artefacts with respect to single JPEG compressed

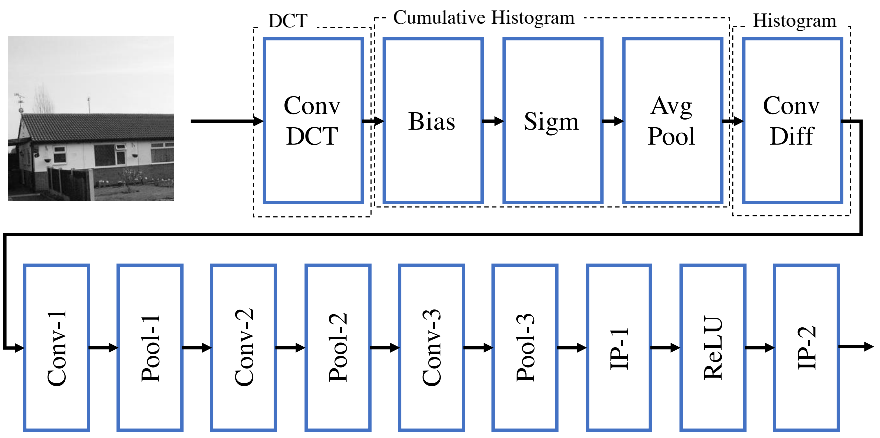


Figure 3.2: *Pipeline of the CNN layers used by the third proposed method. On the top, the part devoted to DCT histogram computation. On the bottom, the CNN described in Table 3.1.*

one [77, 78]. 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[2].

Despite this approach is similar to the one proposed in [13], 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 [13], 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.

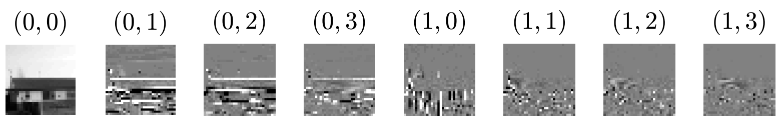
Since this method does not make use of any pre-processing operation, $\tilde{\mathbf{I}}_n = \mathbf{I}_n$. Then, the used CNN can be thought as split into two parts as show in Figure 3.2: i) the former computes DCT coefficients histograms; ii) the latter, fed with these histogram, is the CNN described in Table 3.1, 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 $\mathbf{D}_{c_1,c_2}$ as the $\frac{L}{8} \times \frac{L}{8}$ 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:

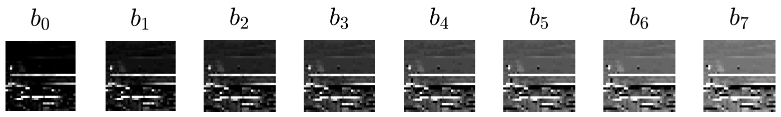
$$\mathbf{D}_{c_1,c_2} = \text{conv}_8(\mathbf{I}, h_{c_1,c_2}), \tag{3.3}$$

where $\text{conv}_8(\cdot,\cdot)$ computes the valid part of the 2D linear convolution using stride 8, and h_{c_1,c_2} is the DCT base at (c_1, c_2) frequency. An example of $\mathbf{D}_{c_1,c_2}$ is reported in Figure 3.3.

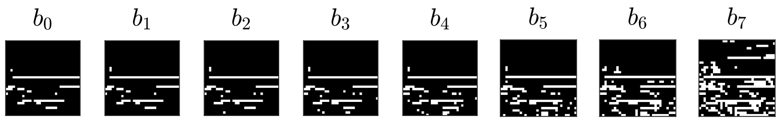
[2]We do not consider the case in which the image in the DCT domain is directly fed to the CNN, because, based on some preliminary experiments, we did not obtain very good performances on small images ($L = 64$).



(a) $\mathbf{D}_{c_1,c_2}$



(b) $D_{0,1} - b$



(c) *sigmoid*$(\gamma \cdot (D_{0,1}(i,j) - b))$

Figure 3.3: *Outputs of CNN layers devoted to histogram computation: (a) output of the DCT layer* $\mathbf{D}_{c_1,c_2}$ *for nine different pairs* (c_1, c_2)*; (b) output of the bias layer* $\mathbf{D}_{c_1,c_2} - b$ *for* $(c_1, c_2) = (0, 1)$ *and different* b *values; (c) output of sigmoid layer sigmoid*$(\gamma \cdot (\mathbf{D}_{c_1,c_2}(i,j) - b))$ *for* $(c_1, c_2) = (0, 1)$ *and different* b *values.*

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 $B_{c_1,c_2}(b)$ of values in $\mathbf{D}_{c_1,c_2}$ that are greater than a constant b, we resort to a series of bias, sigmoid and average-pooling layers obtaining:

$$B_{c_1,c_2}(b) = \frac{L^2}{64} \sum_{i,j \in [0,7]} \text{sigmoid}\left[\gamma \cdot (\mathbf{D}_{c_1,c_2}(i,j) - b)\right], \tag{3.4}$$

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 $\mathbf{D}_{c_1,c_2}(i,j) - b$ (i.e., to obtain very high values for $\mathbf{D}_{c_1,c_2}(i,j) > b$ and very low values for $\mathbf{D}_{c_1,c_2}(i,j) < b$), 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 $\frac{L^2}{64}$. In other words, $B_{c_1,c_2}(b)$ is the b-th cumulative histogram bin for DCT coefficient (c_1, c_2). Examples of these signals are depicted in Figure 3.3.

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

$$Z_{c_1,c_2}(b) = \text{conv}_1(B_{c_1,c_2}, [1, -1]), \tag{3.5}$$

where $\text{conv}_1(\cdot, \cdot)$ computes 1D convolution, and the filter $[1, -1]$ acts as differentiator in the b-th direction. Differently from [13], 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 and must be sought.

Once all histograms 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, 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 3.2) can be considered as an image, fed as input to the CNN pipeline defined in Table 3.1.

3.3.3 Experimental Setup

In this section we report all the details about experimental setup used to evaluate the proposed techniques.

Dataset Construction

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 QFs, for a total amount of more than 3 million images. All datasets are built starting from images of RAISE database [92]. 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) $L = 64, 256$; 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 ($\mathcal{H}_0$), images of size $L \times L$ are single compressed with QF QF; for the second class ($\mathcal{H}_1$), double compressed images are built by coding $L \times L$ images first with various QF1 and then with QF2. For a meaningful analysis, we take QF $=$ QF2 as done in [13].

To build double compressed images for the non-aligned case, we start from images of size $L' \times L'$ with $L' \geq L+7$. 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 $L \times L$, 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 3.2 reports the breakdown of all these training datasets. We denote with $\bar{\mathcal{D}}$ datasets for the A-DJPEG case and with $\hat{\mathcal{D}}$ datasets for NA-DJPEG scenario. Superscripts indicate the adopted QF2 (i.e., 75, 85 or 95), whereas subscripts indicate image size (i.e., $L = 64$ or $L = 256$).

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 $(\text{QF1}, \text{QF2})$ 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 $L = 64$ and $L = 256$, 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 $(\text{QF1}, \text{QF2})$ pairs used for training. Then, to test their generalization capability, we also measure the performance of the detectors with respect to $(\text{QF1}, \text{QF2})$ pairs never used for training.

Table 3.2: *Datasets used for training. All datasets are balanced in both classes and QF pairs.*

Datasets	I Size	QF1	QF2	Alignement	# Train	# Val.
$\bar{\mathcal{D}}_{256}^{(75)}/\bar{\mathcal{D}}_{256}^{(85)}$	256×256	$50, 60, 70, 80, 90$	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	-	-
$\bar{\mathcal{D}}_{64}^{(75)}/\bar{\mathcal{D}}_{64}^{(85)}$	64×64	$50, 60, 70, 80, 90$	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	-	-
				Tot Images	3 480k	360k

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 [53] 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. As best CNN trained model, we always selected the one at the epoch with minimum validation loss, to avoid overfitting.

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 C_{pix} to refer to the CNN-based detector in the pixel domain, C_{noise} for the one in the noise domain, and C_{hist} for the case of CNN embedding DCT histogram computation. Concerning parameters of the latter, we made use of all the AC DCT frequencies. Histograms have been computed using 101 integer bins initialized with $b \in [-50, 50]$.

3.3.4 Results and Discussion

In this section we evaluate the performance of the proposed detectors relying on C_{pix}, C_{noise} and C_{hist}, and we compare them with the state-of-the-art methods. We first focus on the classification in the aligned double JPEG compression scenario, then we move to the case of non-aligned double JPEG compression. Finally, we provide some results in the mixed scenario of aligned and non-aligned double compression.

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

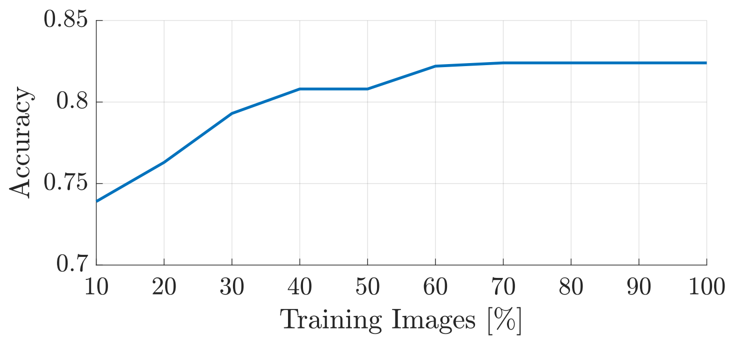


Figure 3.4: *Impact of training set size on A-DJPEG detection accuracy using C_{pix}.*

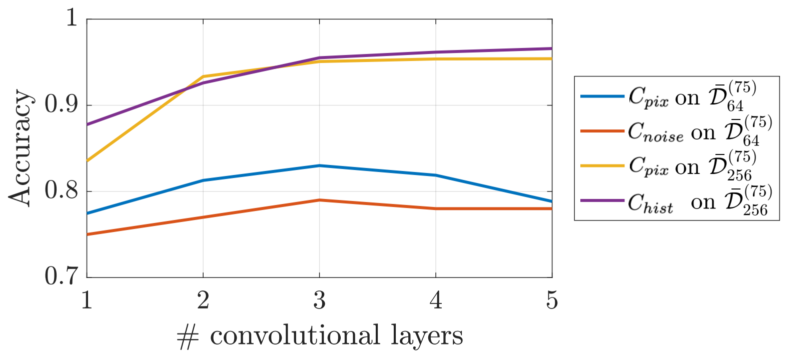


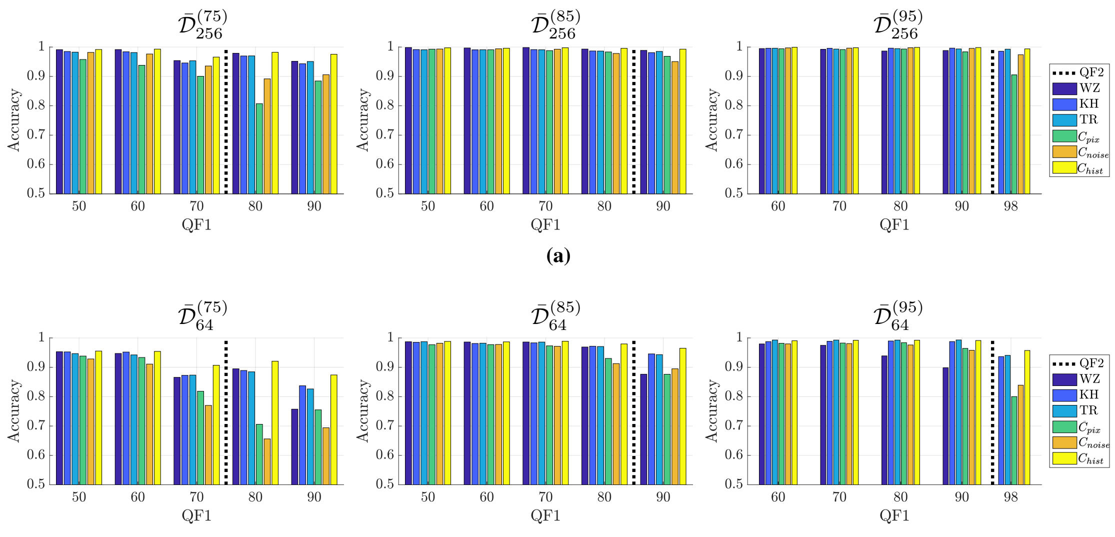
Figure 3.5: *Impact of CNN depth on A-DJPEG detection accuracy using C_{pix} and C_{noise}.*

case, Figure 3.4 shows the results achieved with C_{pix} in the most difficult scenario with small patches ($L = 64$) 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.[3] 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 3.5 shows 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 [13], [71] and [72], denoted respectively as WZ, KH and TR in plot legends. We select [13] as one of the baseline for two reasons: i) it is shown to outperform previously existing state-of-the-art detectors, e.g., [69,70,75,76]; ii) to the best of our knowledge, it is the only method based on CNNs, thus being a natural yardstick for our methods.

Figure 3.6 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 $L = 256$ show that the proposed C_{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 [13] are very distinctive, especially when large images are concerned.

[3]It is worth pointing that the other proposed solutions, i.e., C_{noise} and C_{hist}, usually need less training images to converge.



(a)

(b)

Figure 3.6: *A-DJPEG compression detection accuracy against baselines WZ [13], KH [71], and TR [72]. Dashed black line indicates the considered QF2. On top of each figure we denote the training dataset.*

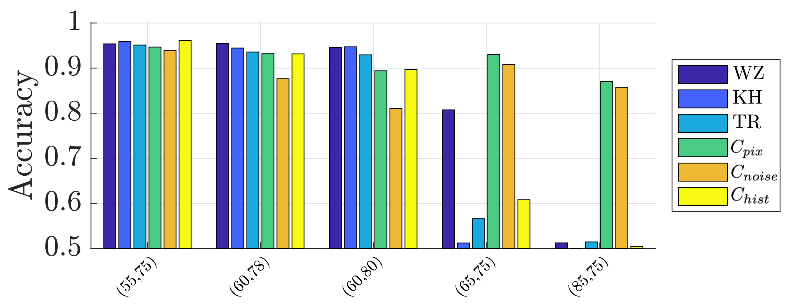


Figure 3.7: *Sensitivity analysis for A-DJPEG compression detection when QF2* $= 75$. *Image size is* 64×64.

With small patches ($L = 64$) all algorithms suffer when QF2 $\cong$ QF1 (this case is addressed in the literature by specific methods tailored for the purpose, e.g., [93]) and QF2 $<$ QF1, as a stronger second compression tends to mask artefacts left by the first one. However, on 64×64 patches, C_{hist} is the one with the best performance and always outperforms state-of-the-art methods on average. Concerning the proposed methods, C_{hist} always outperforms C_{pix} and C_{noise}. This is also expected, as A-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 C_{pix} and C_{noise} allow to achieve accuracy greater than 0.70 on small images.

Regarding generalization capability, Figure 3.7 shows the accuracy achieved by all CNNs trained on the most difficult scenario with QF $= 75$ and small images ($L = 64$). The methods based on DCT histograms or Benford law (i.e., C_{hist} and baselines WZ, KH, TR) suffer to recognize A-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., C_{pix} and C_{noise}) show greater robustness to changes in $(\text{QF1}, \text{QF2})$. To further explore this fact, Table 3.3a

Table 3.3: *Sensitivity of C_{noise} to variations of QF1 and QF2 for A-DJPEG detection. For any pair, only one between QF1 and QF2 is common to images used in the training set (reported in **bold**).*

(a) *Train on* $\bar{\mathcal{D}}_L^{(75)}$, $L \in \{64, 256\}$.

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

(b) *Train on* $\bar{\mathcal{D}}_L^{(85)}$, $L \in \{64, 256\}$.

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

shows the behavior of C_{noise} trained on $\bar{\mathcal{D}}_{64}^{(75)}$ and $\bar{\mathcal{D}}_{256}^{(75)}$ and tested on images with several different (QF1, QF2) pairs (similar results hold for C_{pix}). Similarly, Table 3.3b reports the accuracy results with C_{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 [13] and C_{hist}) allow to achieve the best accuracies, the ones based on the analysis of the pixel image (i.e., C_{pix} and C_{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.

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 non-aligned case. In this case, not surprisingly, the algorithm in [13] (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 artefacts shows up in the DCT histograms). Therefore, we select the well-known algorithm for NA-DJPEG detection proposed in [12], denoted as BP, as additional baseline in this case.

Figure 3.8 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 C_{hist} shows poor performance with respect to C_{pix} and C_{noise}. Indeed, similarly to [13], the traces in the DCT domain that C_{hist} looks at are weak in the non-aligned case. On the other hand, CNNs designed to work in the pixel domain show

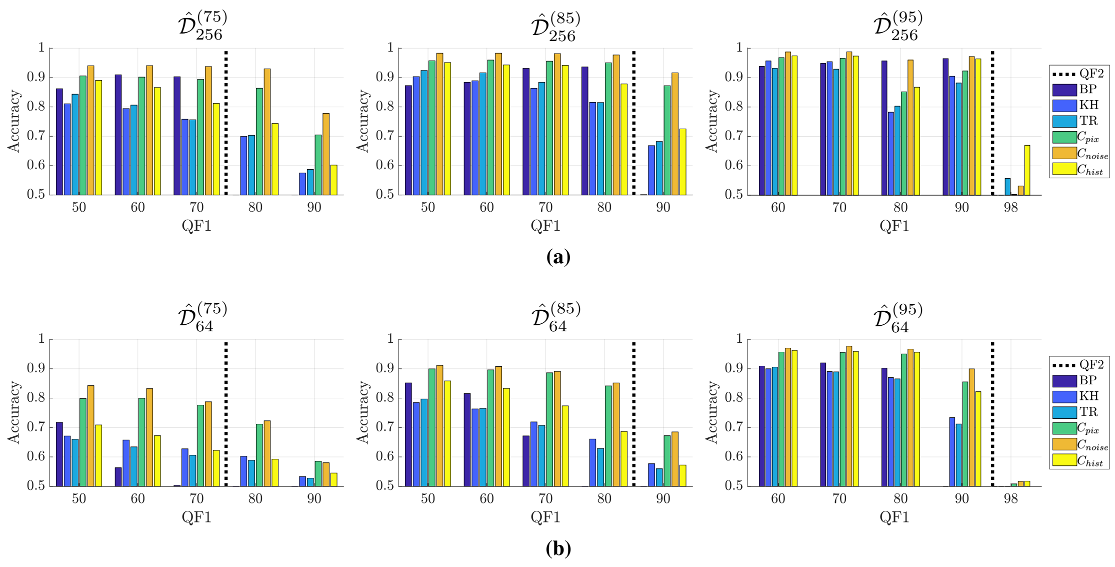


Figure 3.8: *NA-DJPEG compression detection accuracy against baselines BP [12], KH [71], and TR [72]. Dashed black line indicates the considered QF2. On top of each figure we denote the training dataset.*

good detection performance even for small images (i.e., 64×64). From these results, we see that the detector based on C_{noise} always outperforms state-of-the-art.

Concerning network sensitivity to QF pairs different from those in the training set, Table 3.4 shows the results obtained with our best method C_{noise} for both QF1 $= 75$ and 85, and image sizes. As for the aligned scenario, C_{noise} enables good detection accuracy, the only critical cases being those with much higher QF2. It is interesting to notice that C_{noise} is able to detect NA-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 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.

Aligned and non-aligned 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 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 ($L = 64$). Figure 3.9 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, C_{hist} tends to learn better characteristics of A-DJPEG and performs poorly in non-aligned case. Conversely, C_{pix} and C_{noise} are more stable solutions being

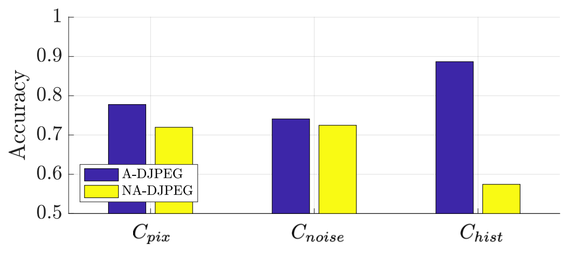
Table 3.4: *Sensitivity of C_{noise} to variations of $QF1$ and $QF2$ for NA-DJPEG detection. Test and training images have only $QF1$ or $QF2$ in common (reported in* ***bold****).*

(a) *Train on* $\hat{\mathcal{D}}_L^{(75)}$, $L \in \{64, 256\}$.

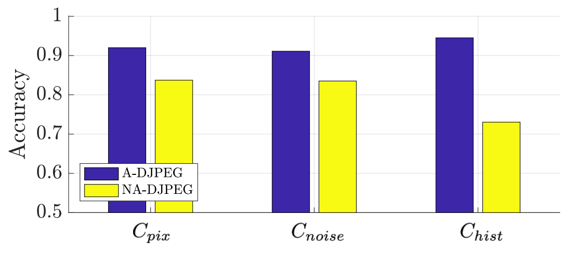
Testing (QF1, QF2)	$L = 64$	$L = 256$
$(55, \mathbf{75})$	0.816	0.876
$(65, \mathbf{75})$	0.805	0.866
$(75, \mathbf{75})$	0.764	0.842
$(85, \mathbf{75})$	0.674	0.776
$(\mathbf{60}, 78)$	0.777	0.845
$(\mathbf{70}, 78)$	0.765	0.830
$(\mathbf{60}, 80)$	0.723	0.794
$(\mathbf{70}, 80)$	0.720	0.790

(b) *Train on* $\hat{\mathcal{D}}_L^{(85)}$, $L \in \{64, 256\}$.

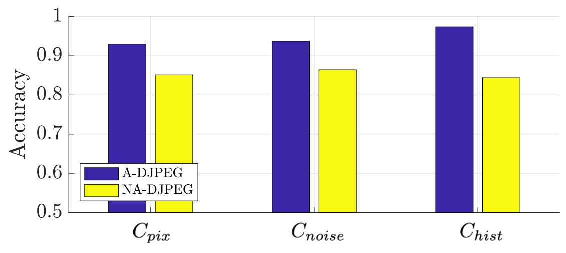
Testing (QF1, QF2)	$L = 64$	$L = 256$
$(55, \mathbf{85})$	0.897	0.972
$(65, \mathbf{85})$	0.878	0.972
$(75, \mathbf{85})$	0.865	0.961
$(85, \mathbf{85})$	0.793	0.954
$(\mathbf{70}, 88)$	0.751	0.786
$(\mathbf{80}, 88)$	0.738	0.785
$(\mathbf{70}, 90)$	0.650	0.610
$(\mathbf{80}, 90)$	0.634	0.600



(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 3.9: *DJPEG compression detection accuracy tested separately on aligned and non-aligned cases, when training is performed on a mixed dataset. Image size is 64 and QF2* $= \{75, 85\}$.

able to detect with almost the same accuracy both A-DJPEG and NA-DJPEG images.

Driven by the accurate performance of C_{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

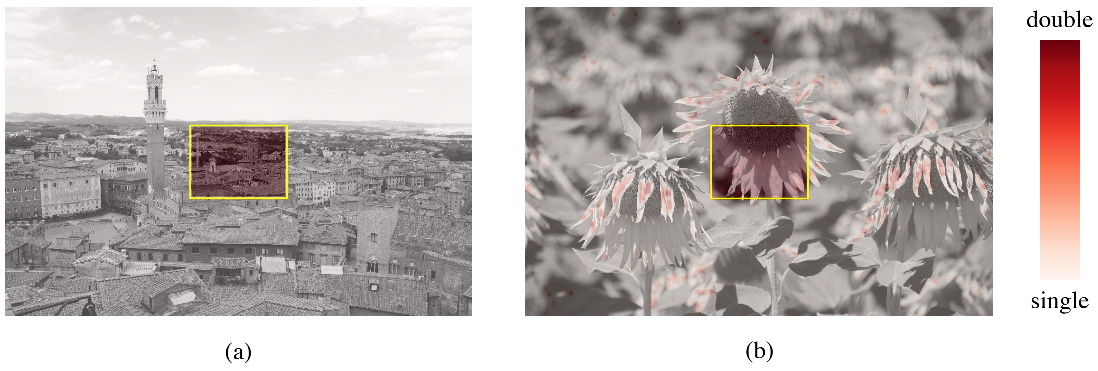


Figure 3.10: *A-DJPEG (a) and NA-DJPEG (b) localization example of a compressed central region with $QF2 = 95$ and $QF1 = 80$. C_{hist} is used for the aligned case while C_{noise} for the non-aligned case. Actual forged region lies inside the yellow rectangle.*

output provided by C_{hist} trained on A-DJPEG images, and the output of C_{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.

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 C_{noise}) and the softmax output is used as an estimation of the probability that the block is double compressed. Figure 3.10 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 C_{hist} is used for the former case and C_{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.

3.4 Multiple JPEG compression detection

We propose a method for detecting multiple-JPEG compressions. This means, given an image, detect how many times (up to four) it has been JPEG compressed. To do so, we leverage perturbations of DCT histograms that capture traces of multiple compressions and train a supervised classifier to discriminate between images compressed different amounts of times.

More in details, multiple-JPEG detection is performed using Task-driven Non-negative Matrix Factorization (TNMF) algorithm [94–96]. This method reduces feature dimensionality, which helps avoid data redundancy, by jointly estimating a dictionary for reduced data representation and a multinomial classifier for multiple-JPEG detection. The rationale behind this choice is that, by optimizing the feature dimensionality

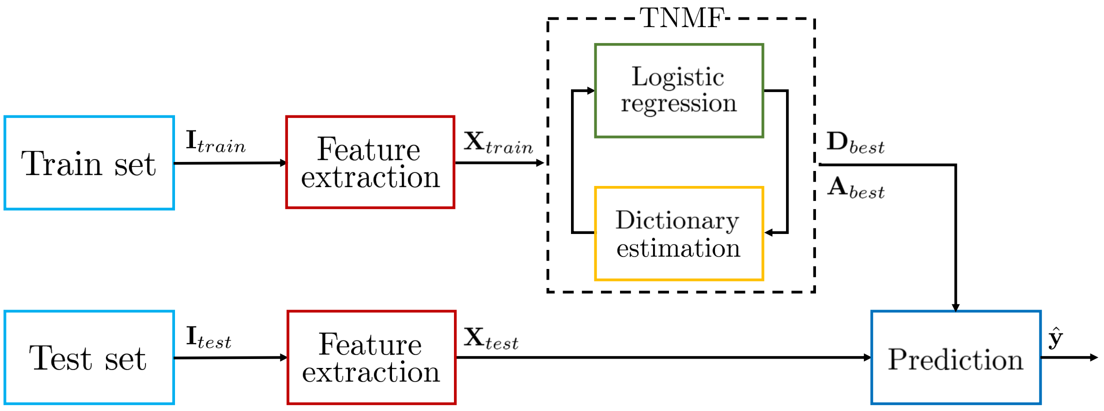


Figure 3.11: *Schematic representation of proposed pipeline.*

reduction method, we should be able to obtain better performance than methods that exploit the FDs as DCT histogram reduction methods [84].

3.4.1 Proposed Method

The proposed pipeline for multiple-JPEG compression classification is depicted in Figure 3.11. During training: (i) a feature vector is extracted from training images; (ii) TNMF algorithm is used in order to jointly learn a dictionary for feature reduction and estimate parameters of a supervised classifier. When the system is trained, a new image can be tested. To this purpose: (i) a feature vector is extracted from the image; (ii) features are projected into a reduced dimensionality space using the learned dictionary; (iii) classifier is used to detect the number of compressions. In the following, we provide a detailed analysis of each step of the algorithm and a simplified example of how TNMF dimensionality reduction works.

Feature extraction

In order to start our analysis, we first extract a set of selected features from each image. We opted for the feature extraction pipeline presented in [13], which exploits block-wise DCT histograms of the image. Multiple compression stages are well known to strongly condition the histograms of DCT coefficients, hence justifying our approach. In particular, the set of investigated coefficients includes only the first 9 AC spatial frequencies taken in zig-zag order. Our choice comes from the more regular trend of lower frequencies coefficients and from the reduced statistics of higher components, which are often quantized to zero [84]. For what concerns the histograms, we select only the first 21 central bins for each DCT band, ending up with a feature vector $\mathbf{x} \in \mathbb{R}^n_+$ of n elements per image, with $n = 189$. Notice that $\mathbf{x}$ assumes non-negative values only.

TNMF Training

Since we aim at classifying multiple compressed images, we propose to exploit multinomial logistic regression, in a one-vs-rest implementation. This means that the multi-class classifier is actually composed by four binary classifiers (e.g., one compression vs. others, two compressions vs. others, etc.), and results of these are merged. For each class label, we train the classifier by minimizing the logistic loss function, defined as $\mathscr{L}_s = \mathscr{L}_s(\mathbf{A}, y_i)$, where $\mathbf{A} = \{\mathbf{w}, \mathrm{c}\}$ is the parameter configuration related to that class [97] and y_i is the image label.

In particular, we propose to exploit TNMF, which is capable of finding sparse data representations by learning a dictionary suited to the specific task of classification [95]. TNMF model allows to learn the task-driven dictionary and the classifier parameters in a joint iterative fashion. More specifically, we estimate a classifier which is optimized with respect to standard logistic regressor, thanks to a particular representation of input data: DCT features extracted from images are projected on a dictionary that is actually tailored to our multinomial classification task. The algorithm works iteratively, alternating the updating of classifier and dictionary, until a fixed number of iterations is achieved. It follows an exhaustive illustration of the method.

Feature reduction. At each iteration t, TNMF starts with feature reduction.

First of all, we select as dictionary the one from the previous iteration, hence $\mathbf{D}_t = \mathbf{D}_{t-1}$, with $\mathbf{D}_t \in \mathbb{R}_+^{n \times p}$. Given a training vector $\mathbf{x}_i$ generated from image $\mathbf{I}_i$, TNMF model considers the optimal projections of data points on the dictionary, with the constraints that all the elements of $\mathbf{x}_i$, $\mathbf{D}_t$ and the obtained projections are non-negative. Notice that p corresponds to the desired size of reduced features, n is the input feature size, thus $p < n$.
Typically, the problem is formulated as follows:

$$\mathbf{h}_{i,t}(\mathbf{D}_t) = \arg\min_{\mathbf{h} \in \mathbb{R}_+^p} \; ||\mathbf{x}_i - \mathbf{D}_t\,\mathbf{h}||_2^2 \; + \; \xi_1 ||\mathbf{h}||_1 \; + \; \xi_2 ||\mathbf{h}||_2^2 \,. \tag{3.6}$$

$\mathbf{h}_{i,t}(\mathbf{D}_t) \in \mathbb{R}_+^p$ is the estimated projection, ξ_1 and ξ_2 are regularization penalty terms, in order to impose sparsity (ℓ_1 norm) and to obtain a strongly convex problem (ℓ_2 norm) hence guaranteeing a unique solution. Equation (3.6) can be solved with standard techniques available in the literature as shall be cleared in the experimental results section.

Classifier updating. Once we have defined the optimal projections, we can use them to update the classifier, associating each vector $\mathbf{h}_{i,t}$ to its related class label y_i. This operation is performed by the minimization of the expected value of loss function $\mathscr{L}_s$ over the entire training set:

$$\mathbf{w}_t,\, \mathrm{c}_t = \arg\min_{\mathbf{w},\mathrm{c}} \; \mathbb{E}_{y_i,\mathbf{x}_i}[\mathscr{L}_s(y_i, \mathbf{w}, \mathrm{c}, \mathbf{h}_{i,t}(\mathbf{D}_t))] \; + \; \upsilon ||\mathbf{w}||_2^2 \,, \tag{3.7}$$

where υ is the penalty term of the regularizer, introduced to prevent overfitting in the classifier.

In the proposed framework, minimization is solved by means of the L-BFGS iterative algorithm [98]. In other words, this step consists in training the multi-class logistic regressor exploiting projected training data samples $\mathbf{h}_{i,t}$ and their labels.

Dictionary updating. At the end of each iteration t, the dictionary must be updated considering the trained logistic regressor, thus obtaining $\mathbf{D}_t$ to be used during next iteration $t + 1$. This is done through the minimization of loss function Equation (3.7) with respect to the dictionary $\mathbf{D}_t$. In particular, we update the dictionary by means of SGD [53], evaluating the function in each training sample $\mathbf{x}_i$ and minimizing in an iterative manner. To perform this task, we have to re-evaluate the sparse representation of each training data sample, $\mathbf{h}_{i,t}(\mathbf{D}_t)$, which depends on the dictionary $\mathbf{D}_t$ estimated from already analyzed samples ($\mathbf{x}_j,\ j < i$). To be more specific, the minimization is performed in two sequential steps:

1. We exploit SGD by calculating the gradient with respect to $\mathbf{h}_{i,t}(\mathbf{D}_t)$. Since we work with sparse representations of data, we compute the active set $\mathcal{S}$ by selecting only the indexes $\in \{1, ..., p\}$ for which vector $\mathbf{h}_{i,t}(\mathbf{D}_t) \neq 0$. For the sake of notation, we introduce the variable $\mathbf{g}_{i,t}$, defined as:
$$\mathbf{g}_{i,t} = ([\mathbf{D}_t^\top]_\mathcal{S}[\mathbf{D}_t]_\mathcal{S} + \xi_2 \mathbf{I}_{|\mathcal{S}|})^{-1}[\nabla_{\mathbf{h}_{i,t(\mathbf{D}_t)}} \mathscr{L}_s(y, \mathbf{A}_t, \mathbf{h}_{i,t}(\mathbf{D}_t))]_\mathcal{S}, \tag{3.8}$$
where symbol $[\cdot]_\mathcal{S}$ represents the projection on the active set, and $|\mathcal{S}|$ is the cardinality of $\mathcal{S}$.

2. It follows a projection of $\mathbf{g}_{i,t}$ over the dictionary space, leading to this updating formulation:
$$\mathbf{D}_t = \mathbf{D}_t - \kappa_t(-\mathbf{D}_t\, \mathbf{g}_{i,t}\, \mathbf{h}_{i,t}^\top(\mathbf{D}_t) + (\mathbf{x}_i - \mathbf{D}_t\, \mathbf{h}_{i,t}(\mathbf{D}_t))\, \mathbf{g}_{i,t}) \,. \tag{3.9}$$

To impose the non-negativity of each dictionary element, we select $\epsilon = 10^{-7}$ as floor value in case of negative entries of $\mathbf{D}_t$. The learning rate κ_t is chosen with the same heuristic criterion proposed in [95]: we select it as $\min(\kappa, \kappa \cdot n_{iter}/(10t))$, being κ a parameter to set and n_{iter} the number of iterations.

Given the conspicuous theoretical baggage of TNMF, we skip all the formal derivations, addressing the interested reader to [95] for a thorough explanation. Following the typical framework of dictionary learning problems, we adopt a validation strategy for selecting the best dictionary and classifier. More specifically, we split our data in training and validation set, evaluating the classification accuracy on validation set at each iteration t, and electing as best dictionary the matrix $\mathbf{D}_t$ which returns the best accuracy. We report the pseudo-code of TNMF method in Algorithm 2.

TNMF Testing

Once we estimate the best combination of dictionary and classifier, we are ready for test phase. Given any new image $\mathbf{I}_{test}$, we compute DCT histograms to obtain feature vector $\mathbf{x}_{test}$. By considering the best validation dictionary, $\mathbf{D}_{best}$, we apply Equation (3.6) to project $\mathbf{x}_{test}$ and obtain the reduced feature vector $\mathbf{h}_{test}$. Finally, we feed $\mathbf{h}_{test}$ to the logistic regressor using the best validation configuration $\mathbf{A}_{best}$ in order to perform label prediction, as in a typical classification problem.

TNMF Training: a simplified example

For the sake of simplicity and data visualization, let us consider a simplified problem consisting of a small dataset of images compressed up to three times. From each image we extract feature $\mathbf{x}$ only considering the 7-th DCT frequency, picking the first central 13 bins. Selecting as reduced feature size $p = 3$, we leverage TNMF training algorithm to find a dictionary for representing our data. In particular, as we are working in a simplified scenario aiming at a ternary classification (discriminating up to three JPEG compressions), a good feature reduction method should enable ternary classification in

Algorithm 2 TNMF Training

```
Require: $\mathbf{y}, \tilde{\mathbf{I}}, p, n_{iter}, \xi_1, \xi_2, v$
    Initialize dictionary $\mathbf{D}_0$
    Split training and validation sets:
    $\mathbf{y}_{train},\ \mathbf{y}_{val},\ \tilde{\mathbf{I}}_{train},\ \tilde{\mathbf{I}}_{val}\ \leftarrow\ \mathbf{y},\ \tilde{\mathbf{I}}$
    for $t = 1, \ldots, n_{iter}$
        Initialize dictionary: $\mathbf{D}_t = \mathbf{D}_{t-1}$
        Feature reduction: $\mathbf{h}_t(\mathbf{D}_t) \leftarrow \mathbf{D}_t,\ \tilde{\mathbf{I}}_{train}$
        Classifier updating: $\mathbf{A}_t \leftarrow \mathbf{h}_t(\mathbf{D}_t),\ \mathbf{y}_{train}$
        Validation accuracy: acc $\leftarrow \mathbf{D}_t,\ \mathbf{A}_t,\ \mathbf{y}_{val},\ \tilde{\mathbf{I}}_{val}$
        $\mathbf{D}_{best},\ \mathbf{A}_{best} \leftarrow \mathrm{acc}_{\max}\{\mathbf{D}_t,\ \mathbf{A}_t\}$
        for $i = 1, \ldots, N$
            Single feature extraction: $\mathbf{h}_{i,t}(\mathbf{D}_t) \leftarrow \mathbf{D}_t,\ \mathbf{x}_{train_i}$
            Active set: $\mathcal{S} \leftarrow$ indexes $\in \{1, \ldots, p\} : \mathbf{h}_{i,t}(\mathbf{D}_t) \neq 0$
            Update learning rate: $\kappa_t \leftarrow \min(\kappa, \kappa \frac{t_0}{t})$
            Update dictionary:
            $\mathbf{D}_t = \mathbf{D}_t - \kappa_t(-\mathbf{D}_t\,\mathbf{g}_{i,t}\,\mathbf{h}_{i,t}^\top(\mathbf{D}_t) + \ldots$
                    $+ (\mathbf{x}_i - \mathbf{D}_t\,\mathbf{h}_{i,t}(\mathbf{D}_t))\,\mathbf{g}_{i,t})$
            Impose non-negativity: $\mathbf{D}_t(\mathbf{D}_t < 0) = \epsilon$
        end
    end
    return $\mathbf{D}_{best}, \mathbf{A}_{best}$
```

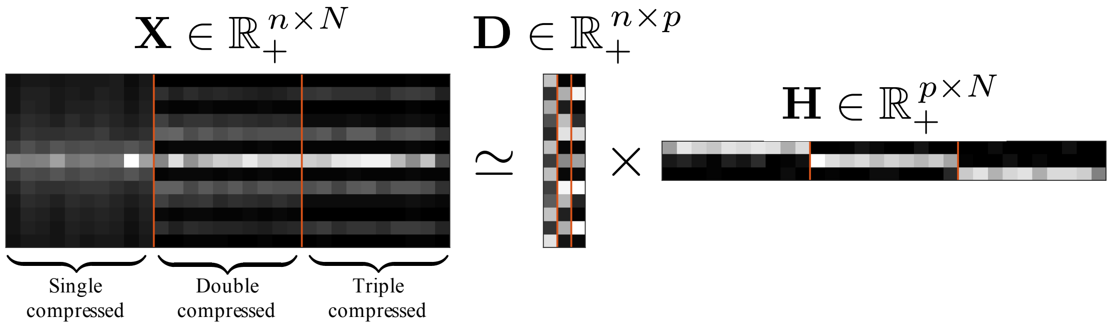

$\mathbf{X} \in \mathbb{R}_+^{n \times N}$
$\mathbf{D} \in \mathbb{R}_+^{n \times p}$
$\mathbf{H} \in \mathbb{R}_+^{p \times N}$
$\simeq$
$\times$
Single compressed
Double compressed
Triple compressed

Figure 3.12: *Simple TNMF example: $\tilde{\mathbf{I}}$ is the matrix of data, specifically N is the total amount of training samples. We can exploit TNMF to approximate $\tilde{\mathbf{I}}$ as the product of the dictionary $\mathbf{D}$ and the matrix $\mathbf{H}$ containing in its columns the optimal projections of $\tilde{\mathbf{I}}$ on $\mathbf{D}$.*

the reduced space. Figure 3.12 depicts the results of dictionary learning through TNMF: we are actually able to estimate a dictionary, associating reduced features to original input data (i.e., classification result is clear just by looking at projected features). Notice that matrix $\tilde{\mathbf{I}}$ illustrates quite well the effects of multiple quantization steps: the more the compression stages, the lower the density of the histogram bins.

3.4.2 Experimental results

Here we report results obtained on two different image databases.

Dataset generation

Following the procedure depicted in [84], we build three datasets starting from 1338 images from UCID [99] (384×512 pixels). Given a final QF $=$ QF$_f \in \{75, 80, 90\}$, we compress each grayscale image up to 4 times. The intermediate QF at compression step $i < f$ is randomly chosen in the interval $[\mathrm{QF}_{i+1} - 12,\ \mathrm{QF}_{i+1} - 5] \cup$

$[\text{QF}_{i+1} + 5, \text{QF}_{i+1} + 12]$ to ensure that QF_i differs from QF_{i+1}. We refer to these datasets as $\mathcal{D}^U_{75}$, $\mathcal{D}^U_{80}$, $\mathcal{D}^U_{90}$, each of them with $4 \times 1338 = 5352$ images. In order to test our approach on a larger scale, we build three further datasets starting with 4000 gray-scale images from RAISE database [92]. Due to the large dimensions of these images, we previously center-crop them to 512×512 pixels and then apply the aforementioned compression pipeline. We obtain $\mathcal{D}^R_{75}$, $\mathcal{D}^R_{80}$, $\mathcal{D}^R_{90}$, each of them with $4 \times 4000 = 16000$ images.

TNMF parameters

We follow a common train-validation-test approach, using 70% of each dataset images for training and the remaining for testing. Training set is further divided in training and validation, following a 90%-10% partition. In particular, as recommended in [95], we initialize the dictionary $\mathbf{D}_0$ by the unsupervised formulation of the problem, leveraging the SPAMS toolbox for computations [100]. The size of $\mathbf{D}_0$ has been chosen as trade off between result quality and computational cost: we set p as the 30% of the DCT length, hence drawing a dictionary $\in \mathbb{R}_+^{189 \times 57}$. Moreover, to improve the convergence speed of the training phase, the proposed method works with a minibatch strategy for the SGD. This basically takes into consideration $n_{batch} > 1$ training samples at each iteration of the inner loop, instead of a single one [100].

Due to the large amount of parameters of TNMF algorithm, we select some specific values for tuning (i.e., iterations $\in \{50, 100, 200, 500\}$, batch size $\in \{200, 400\}$, $\kappa \in \{0.001, 0.005, 0.01\}$, $\xi_1 \in \{0.01, 0.1, 0.5\}$) and perform a grid search to obtain the best accuracy on the validation set. In particular, being TNMF an iterative algorithm, number of iterations has a severe impact on validation accuracy, thus we explore different values until convergence.

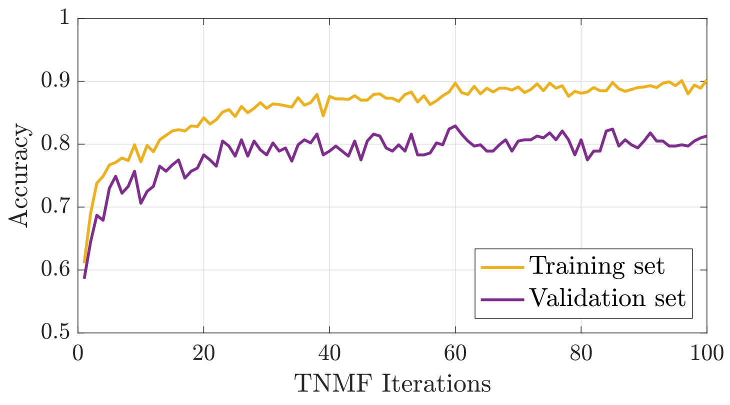
For what concerns the remaining parameters, we set $\xi_2 = 0$ drawing the idea from [95], even though $\xi_2 > 0$ would be necessary for the differentiability of Equation (3.6). This has proven to get satisfactory results in most experiments. The penalty weight in Equation (3.7) is left untouched with respect to the standard formulation of logistic regressor, hence $\upsilon = 1$.

Comparison with state of the art

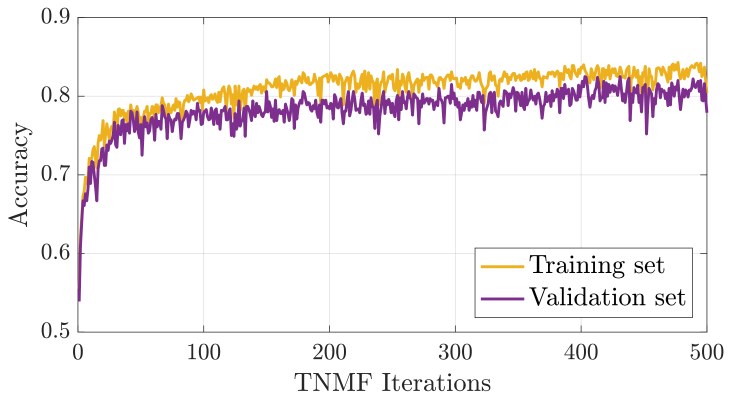
At first glance, we notice that the algorithm needs more iterations on RAISE-derived datasets than on the UCID ones. This is probably due to the huge difference in terms of dataset size.

In this vein, Figure 3.13 depicts the temporal evolution of TNMF accuracy, evaluated for training and validation sets on $\mathcal{D}^U_{75}$ and $\mathcal{D}^R_{75}$. Notice that, if on $\mathcal{D}^U_{75}$ we achieve convergence in at most 100 iterations, $\mathcal{D}^R_{75}$ requires more than 200 iterations. For the sake of brevity, we are showing results for the specific combination of parameters which yields the best validation accuracy, and we will stick to this approach from now on. Nonetheless, we performed a comprehensive investigation on test results for all the parameter configurations, obtaining puny variations among them (standard deviation of test accuracy < 0.01).

Table 3.5 and Table 3.6 show results of the test phase, in terms of mean accuracies and confusion matrices. Specifically, we compare our strategy to [84] (i.e., the only algorithm that deals with four JPEG compressions to the best of our knowledge) and to



(a) *UCID*



(b) *RAISE*

Figure 3.13: *Classification accuracy of TNMF algorithm. (a) UCID Dataset. (b) RAISE Dataset.*

standard logistic regression without the feature reduction step. This last experiment is used to study the actual positive effect of dictionary projection.

Table 3.5: *Mean test accuracies over* 4 *classes for proposed TNMF method, classifier in [84], and logistic regressor (LR) without feature reduction. Best results in bold.*

Dataset	TNMF	[84]	LR
$\mathcal{D}^U_{75}$	$\mathbf{0.78}$	0.73	0.64
$\mathcal{D}^U_{80}$	$\mathbf{0.82}$	0.75	0.65
$\mathcal{D}^U_{90}$	$\mathbf{0.87}$	0.80	0.75
$\mathcal{D}^R_{75}$	$\mathbf{0.80}$	0.76	0.64
$\mathcal{D}^R_{80}$	$\mathbf{0.81}$	0.75	0.64
$\mathcal{D}^R_{90}$	$\mathbf{0.87}$	0.83	0.74

For what concerns the accuracy, our solution is able to go beyond the previously proposed method, since the overall average accuracy (considering all the datasets) is 5.5 percentage points above the mean accuracy of [84].

Regarding the confusion matrices, our results are more accurate than [84] in detection of classes 3 and 4. Indeed, for datasets $\mathcal{D}^R_{75}$ and $\mathcal{D}^R_{80}$ the diagonal terms corresponding to classes 3 and 4 present an average gap of $+0.15$ with respect to state-of-

Table 3.6: *Confusion matrices for $\mathcal{D}^R_{80}$, $\mathcal{D}^R_{90}$. Top: proposed method. Bottom: method in [84]. The highest accuracy among the two methods for a given compression step and dataset is highlighted in yellow.*

$\mathcal{D}^R_{80}$	1	2	3	4
1	**0.980**	0.004	0.015	0.001
2	0.009	**0.850**	0.068	0.073
3	0.079	0.119	**0.711**	0.091
4	0.005	0.158	0.117	**0.720**

$\mathcal{D}^R_{90}$	1	2	3	4
1	**0.997**	0.002	0.001	0.000
2	0.005	**0.952**	0.022	0.021
3	0.022	0.048	**0.833**	0.097
4	0.000	0.126	0.175	**0.699**

$\mathcal{D}^R_{80}$	1	2	3	4
1	**0.999**	0.000	0.000	0.001
2	0.025	**0.960**	0.012	0.003
3	0.182	0.234	**0.523**	0.061
4	0.094	0.275	0.109	**0.522**

$\mathcal{D}^R_{90}$	1	2	3	4
1	**1.000**	0.000	0.000	0.000
2	0.130	**0.863**	0.004	0.003
3	0.029	0.088	**0.828**	0.055
4	0.020	0.179	0.165	**0.636**

the-art, which achieves acceptable multi-classification outcomes especially when QF$_f$ increases up to 90. Concerning the other classes, our results are comparable to [84] for single compressed images, while the detection of double compressed is slightly overwhelmed, probably dictated by a better accuracy of further compressions.

Robustness

In order to preliminary test the method's resilience to editing operations in between JPEG compressions, we applied our detector trained on $\mathcal{D}^R_{90}$ to images that randomly underwent either blurring or gamma correction in addition to compression. This campaign shows that if a single transformation is performed, accuracy drops approximately by 10%. The main effect of the transformations is to hide the previously applied JPEG compression.

3.5 Different JPEG implementations detection

We exploit the rationale behind eigen-algorithms [101] to propose a compact descriptor that captures JPEG implementation traces. Specifically, given a JPEG image under analysis, we re-encode it using different controlled implementations of the JPEG compression algorithm. We then compare the image under analysis with the re-compressed versions to expose salient differences. These differences are collected into a descriptor fed to a simple supervised classifier to detect the JPEG implementation used to encode the image under analysis initially.

3.5.1 Motivations

This section discusses a series of use cases in which detecting traces specific to a JPEG implementation can be helpful from a forensics viewpoint.

Detection of Editing Software

In most situations, tampering with a digital image involves decoding a compressed image in the pixel domain, editing it with one or more software tools, and finally saving it in a compressed format. When the selected compression format is the widespread JPEG standard, different software suites may leverage proprietary solutions to achieve better JPEG compression in terms of bitrate and quality trade-off.

Being able to detect which software has been used (e.g., Photoshop, GIMP) to save an image in JPEG format can be of paramount importance for two different reasons: (i) if the image is recognized to come from an editing suite, then the image cannot be considered pristine with high probability; (ii) if the specific used software can be detected, the list of possible attackers can be narrowed down.

Detection of Device Manufacturer

As for different image editing suites, also different camera manufacturers make use of different JPEG implementations [85]. Even though many robust state-of-the-art detectors are already tailored to camera model attribution problem [33,37,40], capturing and analyzing JPEG traces could be helpful to narrow down the camera model search. First, it could be possible to use traces left by JPEG compression to attribute images to the manufacturer of the camera used to shoot it and then apply a tailored camera model solution to detect the specific model as a refinement step.

Detection of JPEG Anti-Forensics

Forging an image leaving absolutely no traces is not an easy task. Indeed, an expert attacker has to cope with at least two different kinds of traces he might leave: (i) traces left in the pixel domain by processing operations (e.g., multiple compressions, resizing, noise addition, copy-move) that can be captured by one or more of the many passive image forensic tools proposed in the literature [42,66,67]; (ii) traces left in headers and metadata (e.g., missing fields with respect to original images coming from cameras, name of the used image processing suite, incoherent geo-tagging information) that can be easily detected by simple header inspection.

One of the most trivial mistakes an attacker might fall into is to edit a JPEG photograph coming from a specific camera model, then save it using one of the default JPEG quantization matrices provided by the used editing software. Different camera models use different custom JPEG quantization matrices. The quantization matrix is stored in each JPEG file as it is needed at the decoding step. If an analyst detects that the quantization matrix stored in the JPEG header is not compatible with the ones used by the camera manufacturer, then the forgery is easily spotted.

To overcome this issue, an expert attacker has two options: (i) save the edited photograph with any quantization matrix from the considered manufacturer; (ii) save the edited photograph using the same quantization matrix as the original picture. However, the first option may leave traces typical of double JPEG compression [12,69]. Therefore, the second option is the safest solution for the attacker. However, as the attacker does not have access to the manufacturer JPEG implementation, the original JPEG photograph and the edited JPEG pictures will still show different JPEG traces despite using the same quantization matrix.

We show that the proposed descriptor captures traces of this specific anti-forensic operation, hardly detectable otherwise. In other words, we can detect whether an image has been compressed with one (e.g., camera) or another (e.g., editing software) JPEG implementation, regardless of the used quantization matrix.

3.5.2 Proposed Method

In this section, we report how to extract the proposed JPEG-based descriptor $\mathbf{f}$ from a JPEG image $\mathbf{I}$ and how to possibly use it for classification tasks based on JPEG image history.

Descriptor Extraction

Let us consider an image in the pixel domain $\mathbf{I}$. For notation simplicity and without loss of generality, let us consider $\mathbf{I}$ a grayscale image. Let us denote as $\tilde{\mathbf{I}}$ the JPEG encoded version of $\mathbf{I}$ in the DCT domain. In a nutshell, $\tilde{\mathbf{I}}$ is obtained by: (i) splitting $\mathbf{I}$ into non-overlapping 8×8 pixel blocks; (ii) applying JPEG-defined DCT transform to each block; (iii) quantizing each DCT coefficient according to a given quantization matrix and a quantization rule (e.g., rounding, ceiling, etc.); (iv) tiling all quantized DCT blocks back together to obtain a matrix $\tilde{\mathbf{I}}$ having the same size of $\mathbf{I}$. Notice that, from an implementational viewpoint, if an image is already available in compressed JPEG format, its DCT representation $\tilde{\mathbf{I}}$ is already available in the bitstream. Thus there is no need to re-compute $\tilde{\mathbf{I}}$ through DCT transform and quantization.

As JPEG implementations may differ (e.g., due to different quantization rules [85], or even more trivially due to the use of different quantization matrices), we can exploit the eigen-algorithm [101] idea to capture JPEG-based traces. This means we can re-compress image $\mathbf{I}$ using different JPEG implementations and check the differences introduced by each implementation with respect to the original image. Deviations and perturbations introduced by different implementations can characterize the implementation itself. Indeed, it also works aiming at detecting double compression with the same quantization matrix leverage DCT residual statistical changes after multiple JPEG compressions [102]. In the following, we report how to capture these deviations utilizing a compact feature vector using a single and then multiple eigen-algorithms.

Single Eigen-Algorithm. To extract a compact feature vector capturing traces left by a single JPEG implementation, we proceed according to the pipeline shown in Figure 3.14. Let us consider that we have access to a specific JPEG implementation denoted as JPEG$^{\mathrm{a}}$. By applying JPEG$^{\mathrm{a}}$ compression to $\mathbf{I}$ once, we obtain the decoded image $\mathbf{I}_1^{\mathrm{a}}$, and the respective DCT representation $\tilde{\mathbf{I}}_1^{\mathrm{a}}$. By applying JPEG$^{\mathrm{a}}$ compression to $\mathbf{I}$ twice, we obtain the decoded image $\mathbf{I}_2^{\mathrm{a}}$, and the respective DCT representation $\tilde{\mathbf{I}}_2^{\mathrm{a}}$.

We first compare $\tilde{\mathbf{I}}$ and $\tilde{\mathbf{I}}_1^{\mathrm{a}}$ by computing the MSE for each of the 64 DCT coefficients, thus obtaining the 64-element feature vector $\mathbf{f}_1^{\mathrm{a}}$, whose i-th element is defined as:

$$\mathbf{f}_1^{\mathrm{a}}(i) = \frac{1}{K}\sum_{k=1}^{K}\left|\tilde{\mathbf{I}}(i,k) - \tilde{\mathbf{I}}_1^{\mathrm{a}}(i,k)\right|^2,\ i \in [1,64], \tag{3.10}$$

where i is the index of a DCT coefficient, K is the number of 8×8 blocks the image is split into during JPEG compression, and k is the index of an 8×8 block.

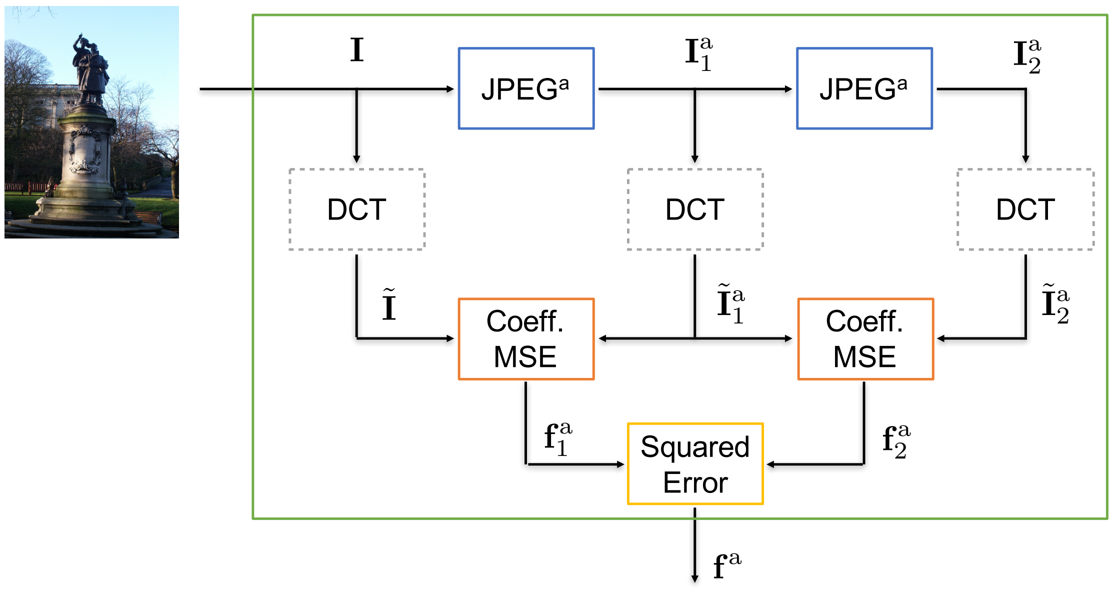


Figure 3.14: *Feature extraction pipeline using a single JPEG implementation (i.e., JPEG^a) as eigen-algorithm.*

Then, we compare $\tilde{\mathbf{I}}_1^{\mathrm{a}}$ and $\tilde{\mathbf{I}}_2^{\mathrm{a}}$ by computing the MSE for each of the 64 DCT coefficients in the same way, thus obtaining the 64-element feature vector $\mathbf{f}_2^{\mathrm{a}}$. Finally, we obtain the full descriptor by computing the element-wise squared error between $\mathbf{f}_1^{\mathrm{a}}$ and $\mathbf{f}_2^{\mathrm{a}}$, thus obtaining $\mathbf{f}^{\mathrm{a}}$, whose i-th element is defined as

$$\mathbf{f}^{\mathrm{a}}(i) = \left|\mathbf{f}_1^{\mathrm{a}}(i) - \mathbf{f}_2^{\mathrm{a}}(i)\right|^2, \ i \in [1, 64]. \tag{3.11}$$

Also $\mathbf{f}^{\mathrm{a}}$ is a 64-element feature vector.

Multiple Eigen-Algorithms. When multiple JPEG implementations are available, we can compute different feature vectors and concatenate them according to the pipeline reported in Figure 3.15. We consider three different JPEG implementations: (i) JPEG$^{\mathrm{a}}$, which uses rounding quantization rule and gives rise to feature vector $\mathbf{f}^{\mathrm{a}}$; (ii) JPEG$^{\mathrm{b}}$, which uses flooring quantization rule and gives rise to feature vector $\mathbf{f}^{\mathrm{b}}$; (iii) JPEG$^{\mathrm{c}}$, which uses ceiling quantization rule and gives rise to feature vector $\mathbf{f}^{\mathrm{c}}$. The three feature vectors obtained with the different implementations are concatenated in a single vector as:

$$\mathbf{f} = \left[\mathbf{f}^{\mathrm{a}},\ \mathbf{f}^{\mathrm{b}},\ \mathbf{f}^{\mathrm{c}}\right]. \tag{3.12}$$

The overall feature vector $\mathbf{f}$ is composed by $64 \times 3 = 192$ elements, and can be used for JPEG-based classification tasks.

Classification

In principle, as $\mathbf{f}$ is a feature vector containing information about JPEG-based image deviations, it can be fed to any supervised or unsupervised learning algorithm. As we are more interested in the amount of JPEG-based information that the proposed feature can capture, rather than in developing a powerful classifier, we use a simple classification technique, i.e., a Random Forest classifier.

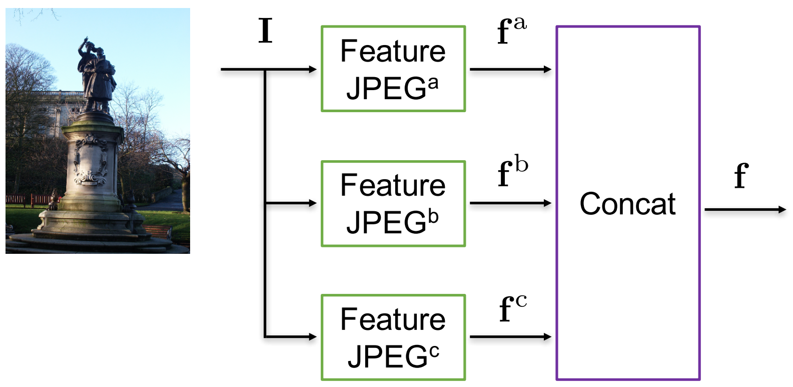


Figure 3.15: *Feature extraction pipeline using multiple JPEG implementations as eigen-algorithms.*

We train a Random Forest classifier without further optimization using feature vectors extracted from a set of training images for all proposed tasks. When a new image has to be classified, we extract the feature vector $\mathbf{f}$ from the image then feed $\mathbf{f}$ to the trained Random Forest classifier. This predicts the likelihood that the image belongs to any class. We select the class with the highest likelihood as the candidate solution. Depending on the application, it is possible to set a custom threshold and only consider images with an estimated likelihood higher than the threshold.

3.5.3 Experiments and Results

In this section we report all the performed experiments and results, separately considering each different use case. A reference implementation of the presented features extractor is available online[4]. All the experiments are carried out on a workstation equipped with Ubuntu 16.04, Python 3.6, PIL 5.2.0, and libjpeg 9.2.0. JPEG compression with Adobe Photoshop CC 2017 is performed on macOS Sierra. It is worth noting that the forensics analyst does not need to know which JPEG implementations are used to compress the images.

As all the experiments involve the use of a supervised classifier (i.e., a Random Forest), we always proceed in the following way: (i) given a dataset, we randomly split its samples into 50% training and 50% testing; (ii) the Random Forest is trained using default parameters[5] on the training set; (iii) results are reported on the test set. As no hyper-parameters tuning is performed, there is no need for a validation set of data. This is done on purpose in order to evaluate the proposed feature discrimination capability with a simple classifier, rather than our ability of fully optimizing a machine-learning technique.

Photoshop vs. PIL (Single Compression)

The first experiment we performed is a two-class classification problem: to detect whether a single compressed JPEG image has been saved using Adobe Photoshop CC 2017 (hereinafter Photoshop) or the Python Imaging Library (PIL), considering that the same quantization matrix has been used. To this purpose, we built a dataset starting

[4]`https://github.com/polimi-ispl/jpeg-eigen`
[5]According to scikit-learn implementation [103].

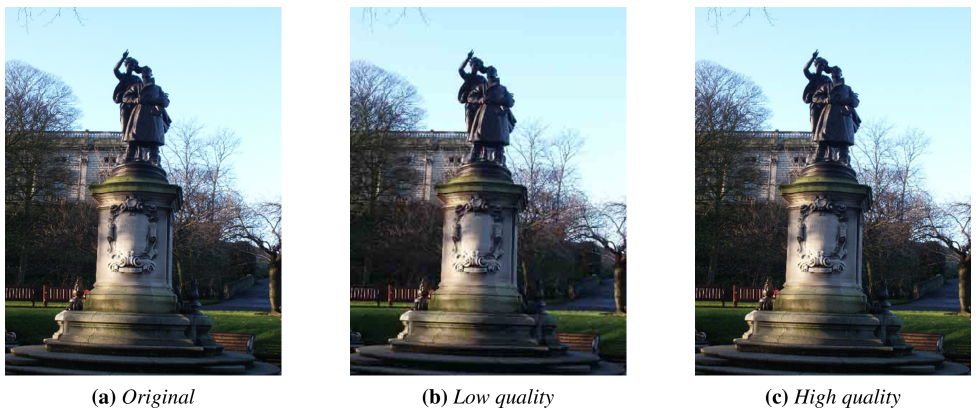


Figure 3.16: *Example of different Photoshop QFs.*

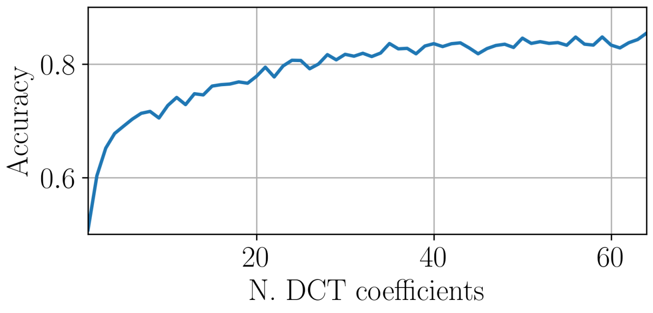


Figure 3.17: *Accuracy for the case $\mathcal{D}^1_{PS}$ vs. $\mathcal{D}^1_{PIL}$ while increasing the amount of used DCT coefficients in zig-zag order.*

from the $1\,338$ uncompressed color images at 512×384 pixel resolution of the UCID dataset [104]. Notice that considering low resolution images makes the problem more challenging. Indeed, less pixels lead to less reliable statistics, thus feature vectors.

We first JPEG compressed each uncompressed image using Photoshop at different JPEG quality levels, thus obtaining four distinct datasets: $\mathcal{D}^1_{\text{PS}}$, $\mathcal{D}^3_{\text{PS}}$, $\mathcal{D}^5_{\text{PS}}$ and $\mathcal{D}^7_{\text{PS}}$ consisting of $1\,338$ single compressed images at quality 1, 3, 5 and 7, respectively[6]. Then, we compressed each UCID uncompressed image with PIL, forcing the use of the respective JPEG quantization matrix used by Photoshop, thus obtaining datasets $\mathcal{D}^1_{\text{PIL}}$, $\mathcal{D}^3_{\text{PIL}}$, $\mathcal{D}^5_{\text{PIL}}$ and $\mathcal{D}^7_{\text{PIL}}$. With this setup, we extracted the proposed feature vector from the luminance component of every picture, and trained a different classifier for each dataset pair (i.e., $\mathcal{D}^1_{\text{PS}}$ vs. $\mathcal{D}^1_{\text{PIL}}$, $\mathcal{D}^3_{\text{PS}}$ vs. $\mathcal{D}^3_{\text{PIL}}$, etc.). Notice that, for these experiments, an image in $\mathcal{D}^1_{\text{PS}}$ and the relative image in $\mathcal{D}^1_{\text{PIL}}$ are single JPEG compressed with the very same quantization matrix. Their differences are only due to the used software, i.e., JPEG implementation.

Figure 3.16 shows an example of original UCID image, the low quality Photoshop version from $\mathcal{D}^1_{\text{PS}}$, and the higher quality Photoshop version from $\mathcal{D}^7_{\text{PS}}$. It is possible to notice that images are not strongly visually degraded by JPEG artefacts.

Figure 3.17 shows the effect on accuracy of using a different amount of DCT coefficients to build the proposed feature vector, considering the scenario $\mathcal{D}^1_{\text{PS}}$ vs. $\mathcal{D}^1_{\text{PIL}}$. It is

[6]Photoshop JPEG quality ranges from 0 (very low) to 12 (very high).

Table 3.7: *Photoshop vs. PIL detection in case of single and double compression. TPR and TNR are the fraction of correctly detected Photoshop and PIL images, respectively.*

(a) *Single Compression*

Task	TPR	TNR	Accuracy
$\mathcal{D}^1_{\text{PS}}$ vs. $\mathcal{D}^1_{\text{PIL}}$	0.89	0.82	0.86
$\mathcal{D}^3_{\text{PS}}$ vs. $\mathcal{D}^3_{\text{PIL}}$	0.82	0.79	0.81
$\mathcal{D}^5_{\text{PS}}$ vs. $\mathcal{D}^5_{\text{PIL}}$	0.80	0.74	0.77
$\mathcal{D}^7_{\text{PS}}$ vs. $\mathcal{D}^7_{\text{PIL}}$	0.77	0.73	0.75

(b) *Double Compression*

Task	TPR	TNR	Accuracy
$\ddot{\mathcal{D}}^1_{\text{PS}}$ vs. $\ddot{\mathcal{D}}^1_{\text{PIL}}$	0.95	0.83	0.89
$\ddot{\mathcal{D}}^3_{\text{PS}}$ vs. $\ddot{\mathcal{D}}^3_{\text{PIL}}$	0.83	0.70	0.77
$\ddot{\mathcal{D}}^5_{\text{PS}}$ vs. $\ddot{\mathcal{D}}^5_{\text{PIL}}$	0.75	0.63	0.69
$\ddot{\mathcal{D}}^7_{\text{PS}}$ vs. $\ddot{\mathcal{D}}^7_{\text{PIL}}$	0.72	0.57	0.65

possible to notice that, by increasingly selecting a greater amount of DCT coefficients read in zig-zag mode, accuracy increases. This is expected, as the more the coefficients, the better the captured JPEG deviations. This validates the idea of using all 64 JPEG DCT coefficients.

Table 3.7a reports results in terms of True Positive Rate (TPR) (i.e., Photoshop images correctly detected), True Negative Rate (TNR) (PIL images correctly detected) and accuracy for each dataset pair. It is possible to notice that, the lower the JPEG quality, the better the results. Indeed, accuracy for JPEG low quality images is higher than 86%, and it drops to 75% when high quality images are considered. This behavior is not surprising, as it is reasonable to assume that low QFs lead to more pronounced artefacts.

Finally, Figure 3.18 shows the receiver operating characteristic (ROC) curve for each dataset pair obtained thresholding the soft output of each Random Forest classifier. Indeed, for a two-class problem, the output of the classifier can be interpreted as the likelihood of an image to belong to a single class. It is possible to notice that by properly selecting this threshold, it is possible to enforce a specific working condition in terms of TPR and False Positive Rate (FPR).

Photoshop vs. PIL (Double Compression)

The second experiment we performed is a more challenging version of the first one: to detect whether an image has been originally JPEG compressed using Photoshop or PIL, given that afterward it is re-compressed with the same quantization matrix using PIL. For this experiment, we took all previously built datasets, and re-compressed each image using PIL and the same quantization matrix used for the first compression. We therefore generated datasets $\ddot{\mathcal{D}}^1_{\text{PS}}$, $\ddot{\mathcal{D}}^3_{\text{PS}}$, $\ddot{\mathcal{D}}^5_{\text{PS}}$, $\ddot{\mathcal{D}}^7_{\text{PS}}$ (i.e., first compression with Photoshop and second with PIL), and $\ddot{\mathcal{D}}^1_{\text{PIL}}$, $\ddot{\mathcal{D}}^3_{\text{PIL}}$, $\ddot{\mathcal{D}}^5_{\text{PIL}}$, $\ddot{\mathcal{D}}^7_{\text{PIL}}$ (i.e., both compressions with PIL). Notice that, each image in $\ddot{\mathcal{D}}^1_{\text{PS}}$ has been double compressed with the same quantization matrix, and the relative image in $\ddot{\mathcal{D}}^1_{\text{PIL}}$ underwent the same processing. The only difference is the software used for the first compression step (i.e., Photoshop or PIL). Also in this scenario, we analyzed each dataset pair separately according to their QFs.

Table 3.7b reports the TPR (i.e., Photoshop images correctly detected), TNR (PIL images correctly detected) and accuracy for each dataset pair. It is possible to notice also this time that the accuracy increases for low JPEG qualities. Indeed, accuracy

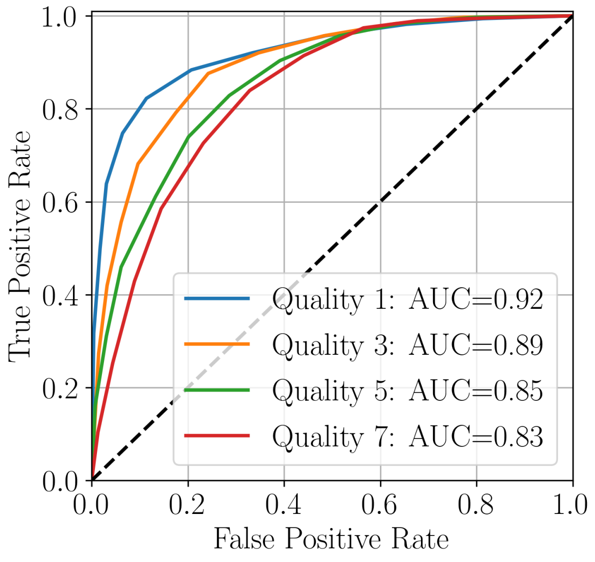


Figure 3.18: *ROC curves showing the different possible working points for each dataset pair according to the used JPEG quality in case of single compression.*

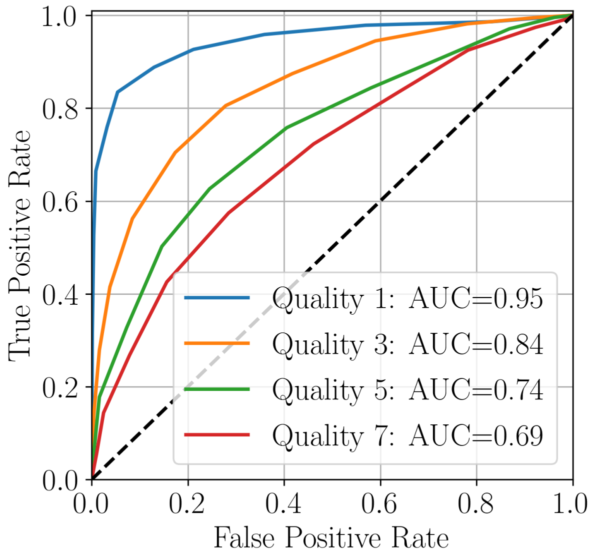


Figure 3.19: *ROC curves showing the different possible working points for each dataset pair according to the used JPEG quality in case of double compression.*

ranges from 89% for low quality images, to 65% for higher quality pictures. However, given the challenging task, we can conclude that the proposed feature vector is still a viable solution toward capturing JPEG implementation traces.

Finally, Figure 3.19 reports the ROC curves obtained thresholding Random Forest outputs for each dataset pair. It is possible to notice that, in case of double compression, low JPEG QFs lead to better results compared to single JPEG compression scenario. Conversely, if two JPEG compressions at high quality are applied, traces of the used software are hindered.

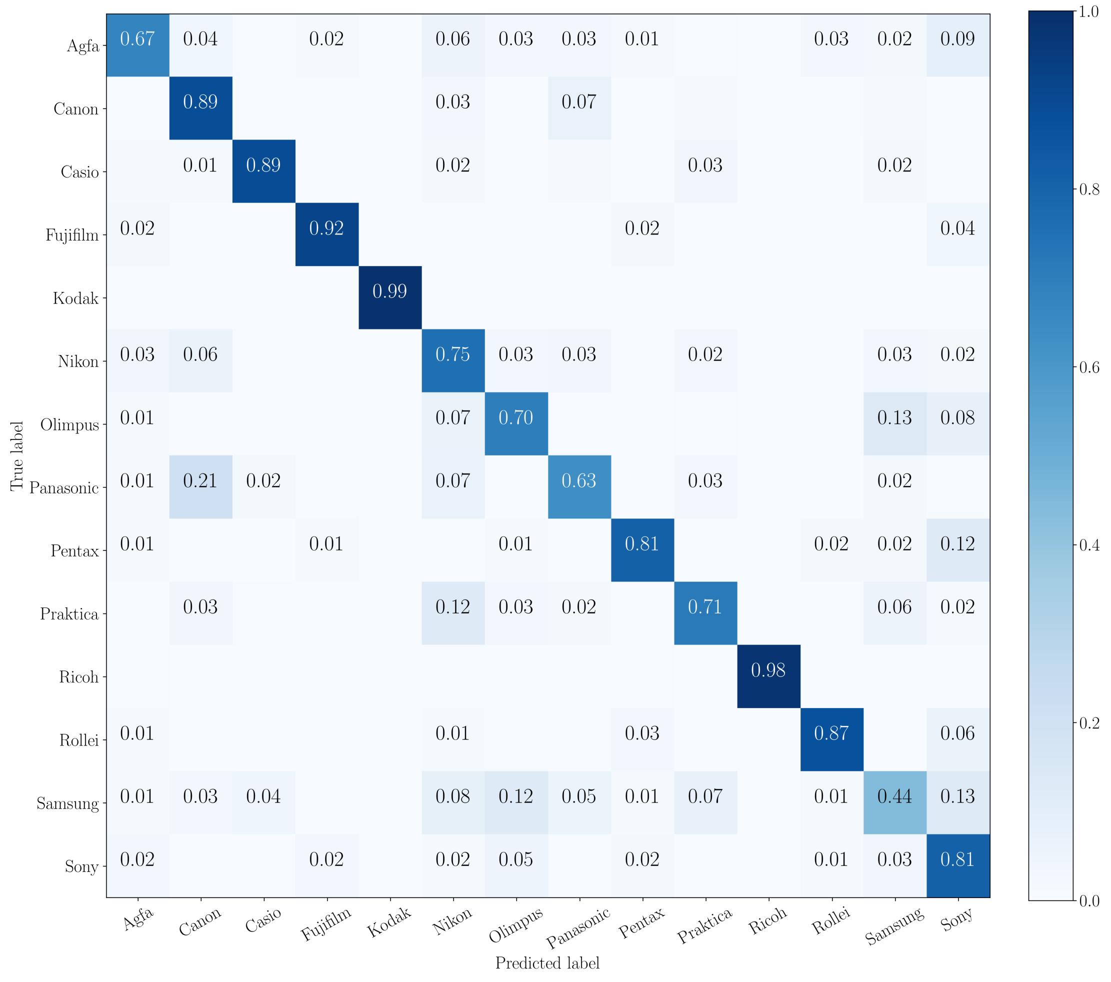


Figure 3.20: *Camera manufacturer detection confusion matrix. Entries smaller than* 1% *are not reported.*

Device Manufacturer

The third experiment we performed can be considered a multi-class classification problem: to detect the manufacturer of the camera used to shot a JPEG image within a closed set of possible known vendors. To this purpose, we selected all images from Dresden Image Dataset [54]. This dataset is composed by more than $16\,000$ images belonging to almost 30 camera models from 14 different camera manufacturers. Figure 3.20 shows the confusion matrix reporting all misclassification errors for the 14 considered camera brands. The overall accuracy in this case is slightly higher than 79%.

Considering this specific experiment, we believe it is important to make an additional consideration. Many state-of-the-art solutions already exist to solve camera attribution problems in all their shapes (e.g., detecting the camera brand, model, instance, etc.) with high accuracy results. However, these methods do not simply rely on JPEG information. Conversely they exploit many kinds of different camera-related traces (e.g., sensor pattern noise, color filter array interpolations, lens aberrations, etc.). Therefore, we still find interesting to show how it is possible to solve camera brand detection problem, just relying on JPEG traces, even with reduced accuracy. Indeed, we

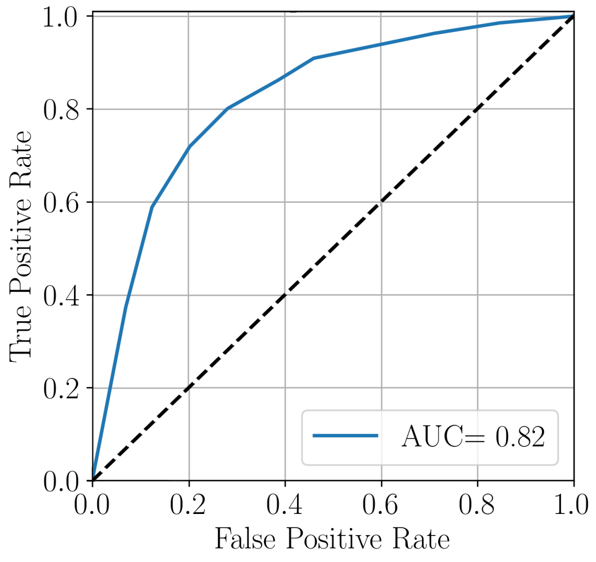


Figure 3.21: *ROC curve obtained on NC2017 dataset for the JPEG antiforensic detection task.*

believe that the proposed feature could be used as additional input to help other camera model detection algorithm in the literature.

JPEG Anti-Forensics

The last experiment we performed consists in solving the following two-class problem related to the JPEG anti-forensic scenario: to detect whether a JPEG image is a pristine one coming directly from a camera, or if it has been edited and re-compressed with the same JPEG quantization matrix. To this purpose, we utilized a state-of-the-art dataset from NIST called Nimble Challenge 2017 (NC2017) for image manipulation detection [105]. This dataset consists of $3\,415$ JPEG compressed images split into two classes: 916 images are pristine, as generated by a camera firmware, and $2\,499$ images are forged. The latter have been JPEG re-compressed with the aforementioned anti-forensic technique, thus a forged image file has the same quantization matrix as the pristine image file that serves as background.

Notice that, as camera models JPEG implementations are not available, two important differences can be observed comparing images belonging to the two classes: (i) forged images are at least double JPEG compressed (if not multiple JPEG compressed); (ii) the JPEG implementation used for the final anti-forensics JPEG compression is most likely different than any JPEG implementation used for pristine images. Figure 3.21 shows the ROC curve obtained by comparing the Random Forest output with different thresholds for the anti-forensic detection task. This shows that it is possible to distinguish pristine images from those edited and re-saved with the JPEG anti-forensic technique with a value of area under the curve of 0.82.

3.6 Conclusions

In this chapter we treated *digital integrity* problems regarding coding traces left on images. 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. The 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 QF is larger than the first one 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. Though this is a common assumption to DJPEG 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 . 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 performances 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%). Future works will be devoted to study fusion techniques to make the most out of each network in a mixed aligned and non-aligned DJPEG case. Moreover, it would be interesting to extend the approach derived in [106] for SVM classifiers, exploiting the idea that robustness to heterogeneous processing and anti-forensics attacks can be recovered by training an adversary-aware version of the classifier.

We also proposed a novel method for detecting multiple JPEG compressions, considering up to four coding steps. Our approach takes advantage of TNMF model, both for feature reduction and for classification, through a joint iterative estimation of dictionary and classifier. We extensively test several setups, taking into account different datasets and QFs. These experiments show that our method outperforms up to date state-of-the-art [84] in terms of classification accuracy.

Finally, we presented a novel descriptor based on the eigen-algorithms idea to capture traces left by different JPEG implementations. The rationale behind the proposed method is that it is possible to analyze a JPEG image under analysis by re-compressing it multiple times with a set of available JPEG implementations. Deviations introduced by eigen-algorithms enable to characterize unknown JPEG implementations, thus allowing to detect the implementation used to compress an image under analysis. The proposed method has been tested considering different challenging use cases. These range from software detection (i.e., Photoshop vs. PIL), to detection of a JPEG anti-forensic technique. Results show that the proposed solution achieves high accuracy on several different tasks, and that it benefits from low quality JPEG compression factors.

Moreover, the proposed feature vector can be interestingly used to cope with camera brand attribution as well.

CHAPTER 4

Semantic Integrity

Fake news over social media are a modern plague we are constantly facing. One of the most used techniques to create visual fake news is the generation of images or videos depicting something not real. Modern tools like Generative Adversarial Network (GAN) and Deepfakes allow malicious users to modify the semantic of what we see, making very difficult to tell the real content and the forged one apart. Unfortunately, in these situations it is not always possible to verify the *digital integrity* of a media through sensor- or coding-based techniques. For this reason, in this chapter, we treat the *digital integrity* of images and videos from a semantic viewpoint. We first propose a method for detecting images generated by means of Deep Learning techniques, and then we extend the analysis of the task to the detection of generated videos.

The chapter is organized as follows: Section 4.1 provides the state of the art for the detection of synthetically generated images and videos. Section 4.2 introduces the main concepts used in the proposed methods. Section 4.3 describes the proposed method about the detection of GAN generated images. Section 4.4 describes the proposed method about the detection of Deepfakes videos. Finally, Section 4.5 draws common conclusion on the chapter.

4.1 State of the Art

With the advent of modern Deep Learning solutions such as GANs, a new series of image and video editing tools has been made available to everyone (e.g., Recycle-GAN [107], StyleGAN [108], StyleGAN2 [109]). These techniques allow the synthesis of realistic and visually-pleasant artificial images, without resorting to complex Computer-Generated Imagery (CGI) techniques required in the past. Unfortunately, this significant step forward in technology comes at a price. Indeed, everyone can maliciously use GANs to generate very realistic image forgeries to manipulate people's opinion through fake news spreading [51]. To counter this threat, the forensic research community has started to develop a series of techniques that detect fake GAN-generated image [110–112].

All of the strategies mentioned above are among the latest solutions for GAN image detection. However, the CGI detection problem has been extensively investigated in the past multimedia forensic literature [113–115]. It is worth noting that previous methods aimed at exposing some specific CGI inconsistencies and artefacts from some characteristic statistical traces or according to a pre-defined model. These strategies were suggested by the knowledge of the available CGI algorithms that could have been applied to generate the fake image. However, GAN-generated images cannot be related to a well-defined model since each scheme presents its peculiarities depending on the implemented architecture and training process. Indeed, as shown in [111], each architecture may introduce different traces, thus making their generalization a complex task. A detector that has been trained to detect images generated by a specific GAN architecture could not be suitable for a different GAN scheme.

Over the past few years, giant steps forward in automatic video editing techniques have also been made. In particular, great interest has been shown towards methods for facial manipulation [116]. To name an example, it is nowadays possible to efficiently perform facial reenactment, i.e., transferring the facial expressions from one video to another one [117, 118]. These techniques enable to change the identity of a speaker with minimal effort.

Systems and tools for facial manipulations are so advanced that even users without previous experience in photo retouching and digital arts can use them. Indeed, code and libraries that work in an almost automatic fashion are more and more often made available to the public for free [17, 119]. This technological advancement opens the door to new artistic possibilities (e.g., movie making, visual effects, visual arts). On the other hand, unfortunately, it also eases the generation of video forgeries by malicious users.

Fake news spreading and revenge porn are just a few possible malicious applications of advanced facial manipulation technology in the wrong hands. As the distribution of these kinds of manipulated videos indubitably leads to severe and dangerous consequences (e.g., diminished trust in media, targeted opinion formation, cyberbullying), the ability to detect whether a face has been manipulated in a video sequence is becoming of paramount importance [120].

Detecting whether a video has been modified is not a novel issue per se. Multimedia Forensics researchers have been working on this topic for many years, proposing different kinds of solutions to different problems [42,66,121]. For instance, in [122,123] the authors focus on studying the coding history of videos. The authors of [124,125] focus on localizing copy-move forgeries with block-based or dense techniques. In [126,127], different methods are proposed to detect frame duplication or deletion.

All the methods mentioned above work according to a common principle: each non-reversible operation leaves a peculiar footprint that can be exposed to detect the specific editing. However, forensics footprints are often very subtle and hard to detect. This is the case of videos undergoing excessive compression, multiple editing operations at once, or strong downsampling [121]. This is also the case of very realistic forgeries operated through methods that are hard to formally model. For this reason, modern facial manipulation techniques are very challenging to detect from the forensic perspective [16]. Many face manipulation techniques exist (i.e., no unique model explains these forgeries). They often operate on small video regions only (i.e., the face or part of it, and not the full-frame). Finally, these kinds of manipulated videos are typically shared through social platforms that apply resizing and coding steps, further hindering classic forensic detectors performance.

4.2 Background

In this section we provide the theoretical background of a Benford's law, which we further employ in the proposed method for detecting generated images. We also highlight how it has been successfully used in the past for a broad amount of forensics tasks. Finally, we provide some background on the Deepfake generation methods.

4.2.1 Benford's Law

Benford's law, which is also known as First Digit (FD) law or significant-digit law, concerns the statistical frequencies of the most significant digits for the elements of a real-life numerical set. More precisely, the rule states that, given a set of measurements for some natural quantities (e.g., population of cities, stock prices), the statistics of their FD follows the distribution depicted in Figure 4.1 and described by the equation:

$$p(d) = \log_{10}\left(1 + \frac{1}{d}\right), \tag{4.1}$$

where d is the FD in base 10 (the generalized version of this law is presented in Section 4.2.1). This has been empirically-observed over a vast range of natural quantities [128], but it is also possible to prove it in closed form for many exponentially-decreasing probability distributions [129]. It has also been observed that this rule is not well-fitted by FD statistics from altered data: whenever numbers are changed according to some selective strategies, FD frequencies deviate from their theoretical values [130]. As a consequence, this proof has been used as supporting evidence for detecting falsified accounts, fake financial reports, and frauds [131]. This property has been largely exploited in Multimedia Forensics to detect image tampering. In fact, natural image Discrete Cosine Transform (DCT) coefficients can be typically modeled by a Laplacian-like distribution [132], which naturally follow Benford's law, and for this reason, the mentioned rule can be successfully used in image forensic applications [133].

A well-known application of Benford's law in forensics is the study of JPEG compression traces [69]: the authors propose using such rule to verify if an image has been JPEG compressed once or twice. [84] exploits FD's features to detect multiple JPEG compression, also showing robustness against rotation and scaling. [82] addresses the multiple JPEG compression detection problem by means of Benford-Fourier analysis. The same authors also investigate traces of previous hidden JPEG compression in uncompressed images [134].

This rule has also been successfully applied to other forensic problems. In [135], the authors show that it is possible to leverage FD distribution to roughly estimate the

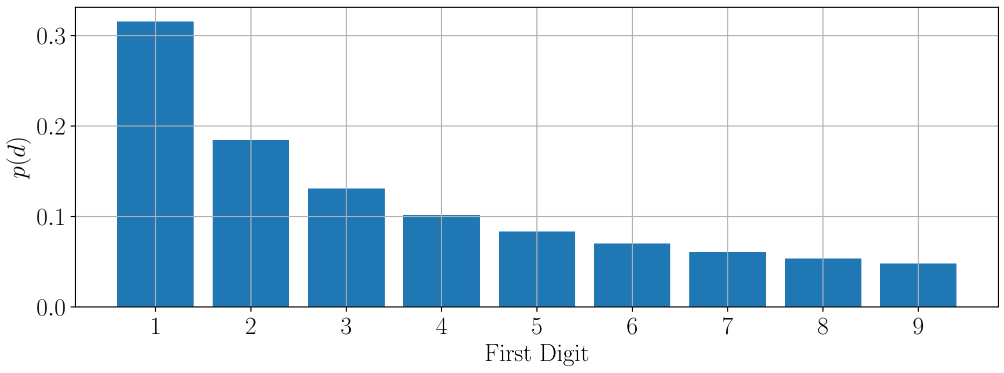


Figure 4.1: *Benford's law first digits Probability Mass Function considering base* 10.

amount of processing that has been applied to a given image. The authors of [136] apply Benford's law to solve image contrast enhancement detection problem. In [137], the authors make use of this law to deal with splicing forgery localization.

Another interesting application of Benford's law in image forensics is detecting computer graphics and computer generated images. To this purpose, [138] models light intensity in natural and synthetic images, concluding that Benford's law is not followed by the latter. [139] shows how to efficiently detect morphed faces using the fitting parameters of the Benford's logarithmic curve as features.

Anyway, detecting synthetic images is nowadays a timely and crucial forensic need due to the achievements of GAN technology in generating highly-realistic fake photographs. This possibility has been recently used to create false image and video contents in deepfake political propaganda, revenge porn, fake news creation. For these reasons, during the last years Multimedia Forensics researchers have been focusing on designing reliable strategies to detect synthetic images.

To this purpose, [110] proposes a method to detect image-to-image translation over social networks. Specifically, the authors compare different detectors fine-tuned for the binary classification task of GAN-generated against natural image detection. The same authors also show how a model-specific fingerprint can be retrieved by GAN generated images in order to identify the specific network used for image generation [111]. In [140], authors apply an incremental learning strategy to train a GAN-generated image detector that can be progressively updated in time as new images from different kinds of GANs are processed. In [141], the authors propose a method to detect GAN-generated images by analyzing the disparities in color components between real scene images and generated images. In [112], GAN images are detected by analyzing saturation artefacts in pixel distributions. Moreover, if videos are analyzed, methods exploiting also the temporal evolution of frames have been proposed [137, 142].

4.2.2 Deepfake videos

The first Deepfake appeared in 2017 on the internet as adult videos where the face of the original subject, usually a female actress, was replaced with some celebrity's face. "Deepfake" is a generic name for the technique that does not refer to a specific algorithm or software. There is no official version of the original code used for the first Deepfake. However, several variants exist together with their implementations.

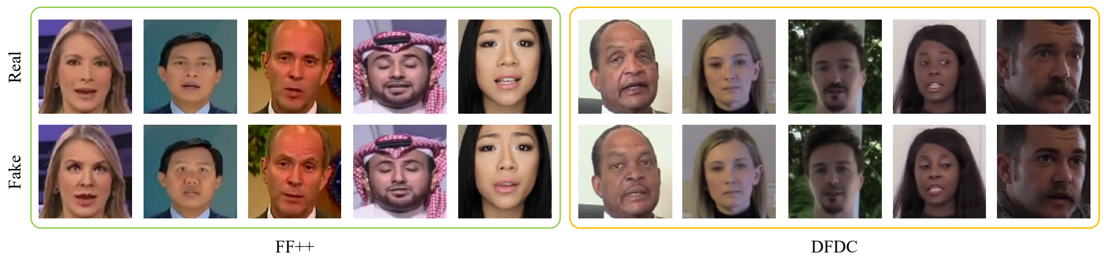


Figure 4.2: *Sample faces extracted from FaceForensics++ (FF++) [145] and DeepFake Detection Challenge (DFDC) [146] datasets. For each pristine face, we show a corresponding fake sample generated from it.*

The most common approach is the Deepfake AutoEncoder (DFAE). This architecture consists of a modified convolutional AutoEncoder (AE) model: one shared encoder has two separately-trained decoders. Each decoder is trained on one of the two identities to swap. This structure makes the encoder learn to encode common features like lighting and pose in the latent space. Conversely, the decoder learns identity-specific features, e.g., eyes, nose shape. To perform the face swap, the original face image is given as input to the shared encoder and then it is passed to the opposite decoder, keeping light and pose information but changing identity features.

An attractive model is Neural Talking Head (NTH) [143]. It can generate talking head models of people from very few or even one single image. This is possible with two training phases: first, meta-learning is performed on a large dataset of videos. After that, a fine-tuning stage uses the meta-parameters to generate realistic results after seeing a couple of target images. Like the DFAE approach, landmarking positions can be extracted from a source video and fed to the generator for another face to perform the face swap Face Swapping GAN (FSGAN) [144] uses GANs to perform a face reenactment and a face swap between a source and a target subject. It comprises several generators: reenactment, segmentation, inpainting and blending. Reenactment and inpainting generators use adversarial loss, while segmentation and Poisson blending are separately trained. Large pose face reenactment generation is handled by collecting face landmarks of source and target and generating intermediate poses. Figure 4.2 shows some examples of Deepfake generated faces from two popular dataset we use in our work.

4.3 Generated image detection exploiting Benford's Law

In this section we present our method for detecting GAN generated image exploiting the statistical trace of Benford's Law.

4.3.1 Motivations

Natural images, as many other natural processes, can be roughly approximated as autoregressive signals [147]. This is the rationale behind different historical as well as more recently proposed image compression [147, 148] and generation [149, 150] methods. From these assumptions, an image can be modeled as a complex autoregressive signal with a generally low-pass characteristics.

GAN generator's structures are usually composed by a concatenation of limited-support convolutional layers followed by non-linearities. Filters' coefficients are optimized so that GAN's response to a given input belongs to the desired output class. However, in most GAN implementations, practical and complexity reasons have led to the adoption of filters with a limited size. Therefore, if no recursive operations are applied in the network architecture, the output of a GAN generator looks more like a signal filtered through a Finite Impulse Response (FIR) filter than a complex autoregressive process.

The rationale behind the proposed method is that the information related to the filter ideally used to generate the data under analysis can be used to discriminate natural images (with autoregressive and complex spectra) from GAN-generated ones (generated through operations closer to FIR filtering). This can be done analyzing the statistics of quantized DCT coefficients.

More precisely, let us assume that an input grayscale image is partitioned into K distinct 8×8 blocks, which are then mapped into the 2D-DCT domain and further quantized. This processing chain is used by the JPEG coding standards and proves to be tailored to the spectral characteristics of images. Some of the past works highlight that, in the frequency domain, the quantized DCT coefficient statistics of natural images must follow Benford's law [133].

Let us denote as $c_{n,\Delta}(k)$ the DCT coefficient at the n-th frequency in zig-zag mode obtained from the k-th block and quantized with step Δ. It is possible to compute the corresponding FD with base b as:

$$d_{b,n,\Delta}(k) = \left\lfloor \frac{|c_{n,\Delta}(k)|}{b^{\lfloor \log_b |c_{n,\Delta}(k)| \rfloor}} \right\rfloor . \tag{4.2}$$

As $d_{b,n,\Delta}(k)$ can only assume $b-1$ values (i.e., all possible digits defined in base b apart from zero), its Probability Mass Function (pmf) $\hat{p}_{b,n,\Delta}$ computed over the K blocks is composed by $b-1$ elements. For the sake of notational compactness, let us momentarily drop the indexes n, b, and Δ. We can formally define the pmf $\hat{p}(d)$ as:

$$\hat{p}(d) = \frac{1}{K} \sum_{k=1}^{K} \mathbf{1}_x(d(k)),\ d \in \{1, 2, \ldots, b-1\}, \tag{4.3}$$

where:

$$\mathbf{1}_x(y) = \begin{cases} 1 & \text{if } y = x, \\ 0 & \text{otherwise.} \end{cases} \tag{4.4}$$

This pmf for a natural image must follow the generalized Benford's law equation:

$$p(d) = \eta \log_b \left(1 + \frac{1}{\upsilon + d^{\omega}} \right), \tag{4.5}$$

where η is a scale factor, υ and ω parameterize the logarithmic curve, and $d \in \{1, 2, ..., b-1\}$ is one possible value of the considered first digits in base b. The fitness between $\hat{p}(d)$ and $p(d)$ can be measured by some divergence functions such as the Jensen-Shannon divergence $D^{\text{JS}}(\hat{p}|p)$:

$$D^{\text{JS}}(\hat{p}|p) = D^{\text{KL}}(\hat{p}|p) + D^{\text{KL}}(p|\hat{p}), \tag{4.6}$$

which is a symmetrized version of the well-known Kullback-Leibler divergence:

$$D^{\text{KL}}\left(\hat{p}|p\right) = \sum_{d=1}^{b-1} \hat{p}(d) \log \frac{\hat{p}(d)}{p(d)}. \tag{4.7}$$

Since D^{JS} proves to be unstable for biased pmfs, it is possible to use the symmetrized Renyi divergence:

$$D_{\alpha}^{\text{R}}\left(\hat{p}|p\right) = \frac{1}{1-\alpha}\left(\log S_{\alpha}\left(\hat{p},p\right) + \log S_{\alpha}\left(p,\hat{p}\right)\right), \tag{4.8}$$

or the symmetrized Tsallis divergence:

$$D_{\alpha}^{\text{T}}\left(\hat{p}|p\right) = \frac{1}{1-\alpha}\left(2 - S_{\alpha}\left(\hat{p},p\right) - S_{\alpha}\left(p,\hat{p}\right)\right), \tag{4.9}$$

where:

$$S_{\alpha}\left(q,p\right) = \sum_{d=1}^{b-1} q(d)^{\alpha}/p(d)^{\alpha-1}. \tag{4.10}$$

It is possible to prove that, whenever an image is altered (e.g., it is compressed/quantized a second time, etc.), Benford's law is not verified anymore. In fact, many modifications redistribute image coefficients among the bins of the quantizer, thus the final pmf associated to quantized DCT coefficients presents some oscillating probability values that deviate from the ideal distribution. For these reasons, many solutions in the past measured the divergence between the empirically-estimated $\hat{p}(d)$ and its ideal fitted version $p(d)$ in order to find whether the image has been altered or not. We show that it is possible to adopt the same solution to detect GAN generated pictures.

4.3.2 Proposed method

We define the GAN-generated image detection problem as a two-class classification problem. Given an image $\mathbf{I}$, we want to understand whether it has been synthetically generated through a GAN, or it is a natural photograph.

Formally, to solve this problem we consider a pipeline composed by two blocks: a feature extractor and a supervised classifier. The feature extractor implements the function $F(\cdot)$, which turns the image into a more compact yet informative representation, i.e., the feature vector $\mathbf{f} = F(\mathbf{I})$. The classifier implements the function $M(\cdot)$ such that: $M(\mathbf{f}) = 0$ if the image is a natural one; $M(\mathbf{f}) = 1$ if the image comes from a GAN. With this framework in mind, we focus on designing the function $F(\cdot)$ based on Benford's law, so that a simple classifier can be effectively used.

The feature extraction process is depicted in Figure 4.3. Given an image $\mathbf{I}$, we divide it in K non-overlapping blocks with resolution 8×8 pixel. From each block, we compute its 2D-DCT representation. We then quantize it using a given quantization step Δ (chosen for each coefficient according to a JPEG quantization matrix).

Given a base b, we compute the first digit of the n-th quantized 2D-DCT frequency sample from the k-th block according to Equation (4.2). We then compute the pmf $\hat{p}_{b,n,\Delta}$ according to Equation (4.3). Examples of $\hat{p}$ for different bases for both natural and GAN-generated images are reported in Figure 4.4. Finally, we fit generalized Ben-

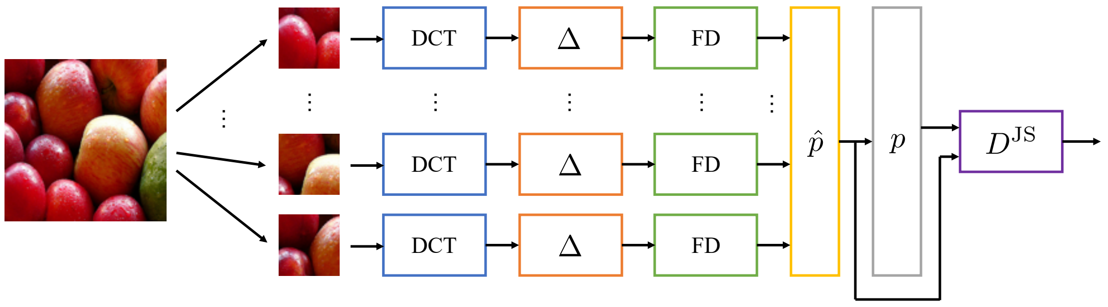
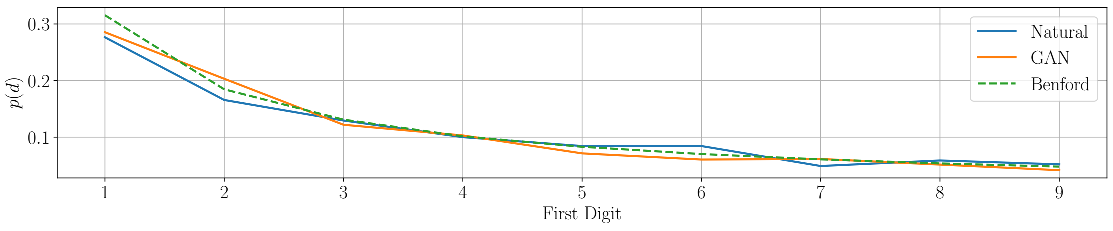
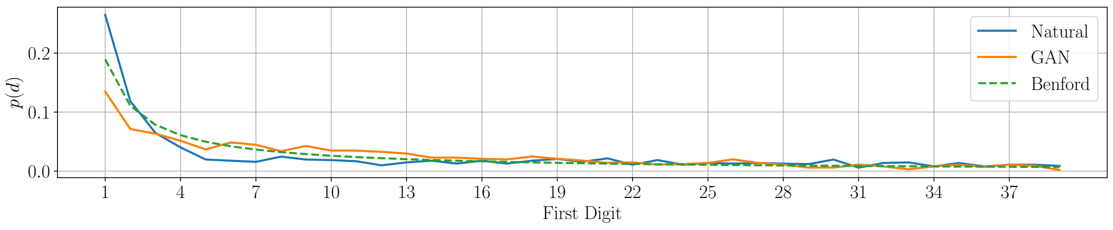


Figure 4.3: *Feature extraction pipeline considering a single divergence, quantization step Δ, base b and DCT coefficient n. Extraction process is repeated for multiple parameters.*



(a) $b = 10$



(b) $b = 40$

Figure 4.4: *Different pmf $\hat{p}$ for natural (blue) and GAN-generated images (orange) are compared to the ideal Benford curve (dashed green) for different bases b. Blue and orange curves deviates differently from the green one.*

ford's law expressed in Equation (4.5) by solving a mean square error minimization problem as:

$$p^{\mathrm{fit}}_{b,n,\Delta} = \arg\min_{p} \sum_{d=1}^{b-1} (\hat{p}_{b,n,\Delta}(d) - p(d))^2. \tag{4.11}$$

Comparing the computed pmf $\hat{p}_{b,n,\Delta}$ and the Benford fit $p^{\mathrm{fit}}_{b,n,\Delta}$, we compute Jensen-Shannon divergence $D^{\mathrm{JS}}_{b,n,\Delta}$, Renyi divergence $D^{\mathrm{R}}_{b,n,\Delta}$, and Tsallis divergence $D^{\mathrm{T}}_{b,n,\Delta}$ as reported in Section 4.3.1. Notice that we removed the dependency of Tsallis and Renyi divergence on α as we keep it constant in our experiments.

Finally, considering a set $\mathcal{B}$ of bases, a set $\mathcal{N}$ of DCT frequencies and a set $\mathcal{J}$ of JPEG Quality Factors (QFs) driving the quantization parameter Δ, we obtain the final feature vector by concatenating all divergences as:

$$\mathbf{f}_{\mathcal{B},\mathcal{N},\mathcal{J}} = [D^{\mathrm{JS}}_{b,n,\Delta},\ D^{\mathrm{R}}_{b,n,\Delta}\ D^{\mathrm{T}}_{b,n,\Delta}]_{b\in\mathcal{B},n\in\mathcal{N},\Delta\in\mathcal{J}}. \tag{4.12}$$

Notice that the feature vector size depends on how many DCT coefficients, bases and quantization steps are used during the analysis. For instance, if we choose a single

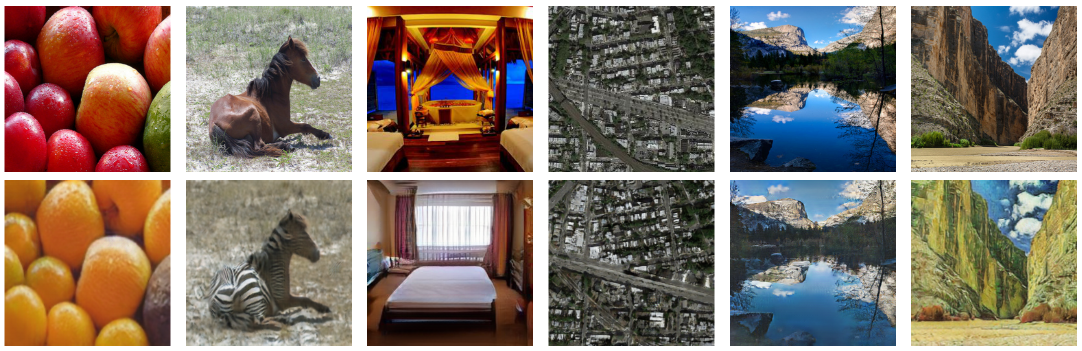


Figure 4.5: *Examples of original (top) and GAN-generated (bottom) images belonging to the dataset proposed in [140].*

compression step, a single DCT frequency and a single base, the feature vector will be composed by the concatenation of just three divergences, thus having dimensionality 3. Conversely, if we use multiple bases, frequencies and compression steps, we end up with a bigger vector. In our experiments, we consider vectors with dimensionality ranging from 3 to 540, as shall be explained in Section 4.3.4.

After feature computation, the vector $\mathbf{f}_{\mathcal{B},\mathcal{N},\mathcal{J}}$ is fed to a supervised classifier. To study the effectiveness of Benford-based features, we do not adopt unnecessarily complicated classifiers $M(\cdot)$. Specifically, we resort to a Random Forest classifier.

4.3.3 Experiments

In this section we describe the considered dataset and all the possible setups we adopt for the experiments.

Dataset

To build our dataset, we started from the publicly available GAN-dataset released by [140]. We considered a corpus composed by 15 different sub-datasets of images obtained employing 2 different architectures: Cycle-Gan [151] and ProGAN [152]. The first architecture is designed for image-to-image translation, i.e., mapping an image of a given class (e.g., pictures of horses) to an image of another one (e.g., pictures of zebras). The second architecture is a generator able of creating natural-looking pictures of different scenes depending on the used training data (e.g., bedroom pictures, bridges, etc.). Each dataset comprises both natural images and their GAN-generated counterparts. All images are color images and have a resolution of 256×256 pixel. The complete composition is reported in Table 4.1 and some examples of the more than $200\,000$ images are reported in Figure 4.5.

Setup

As shown in Section 4.3.2, each feature depends on a selected set of bases $\mathcal{B}$, DCT frequencies $\mathcal{N}$, and analysis JPEG QF describing the set of quantization steps $\mathcal{J}$. With regards to bases, we test all combinations of sets of bases $\mathcal{B} \subseteq \{10, 20, 40, 60\}$ containing from one to four elements. This leads to 15 possible combinations of bases (i.e., from $\mathcal{B} = \{10\}$ to $\mathcal{B} = \{10, 20, 40, 60\}$). Concerning the selected DCT frequencies, we

Table 4.1: *Dataset composition*

Architecture	Dataset	Number of images
Cycle-Gan	orange2apple	1280
	photo2ukiyoe	4072
	winter2summer	1484
	zebra2horse	1670
	photo2cezanne	3978
	photo2vangogh	4099
	photo2monet	4765
	facades	259
	cityscapes	1996
	sats	684
ProGAN	lsun_bedroom	30770
	lsun_bridge	28768
	lsun_churchoutdoor	29120
	lsun_kitchen	42706
	lsun_tower	29020

choose a limited amount of sets $\mathcal{N} \subseteq \{1, 2, \ldots, 9\}$ (i.e., including the first 9 frequencies in zig-zag order after DC) , similarly to other detectors available in literature [84]). Specifically, we only consider 9 sets obtained by progressively-adding one frequency at a time from the previous set (i.e., $\mathcal{N} = \{1\}$, $\mathcal{N} = \{1, 2\}$, ..., $\mathcal{N} = \{1, 2, \ldots, 9\}$). As for the quantization step values, the final feature set was created concatenating the 5 arrays of divergences obtained using JPEG QFs in the set $\mathcal{J} \subseteq \{80, 85, 90, 95, 100\}$. Considering all combinations of bases, frequencies and JPEG quantization steps, we obtain a total amount of 675 different setups.

For each feature vector described in Section 4.3.2 (i.e., each setup), we trained a different Random Forest classifier performing Leave-One-Group-Out cross-validation over the various datasets as explained in [110]. The choice of Random Forest was suggested by the low complexity requirements, the generalization capabilities, and the resilience to small training datasets.

Namely, given a dataset $\mathcal{D}_i$ out of the complete set of dataset $\mathcal{D}$, we trained our model over the remaining $\mathcal{D}_j$, $\forall j \neq i$ and we test over $\mathcal{D}_i$. Results are always shown on the leave-out dataset, and we report the maximum accuracy value among all the different setups. To provide a practical example, let us consider the situation in which we test on dataset $\mathcal{D}_i =$ `orange2apple`. This consists of original images (i.e., apples and oranges) and GAN images (apples turned into oranges and vice versa). The classifier was trained on all the other images (excluding those in `orange2apple`) in order to avoid biasing the results with overfitting. We adopted the Random Forest implementation provided with the open-source Scikit Learn Python library. After a grid search over several candidates, we fixed the number of Decision Trees to 100, with bootstrap sampling enabled. We selected Gini index as splitting policy, leaving all the other parameters as their default values.

To select the baselines, we focused on the work proposed by [110] since, to the best of our knowledge, it is the only work to perform an extensive GAN detection test over a large dataset of images. Specifically, we selected two baselines: a completely

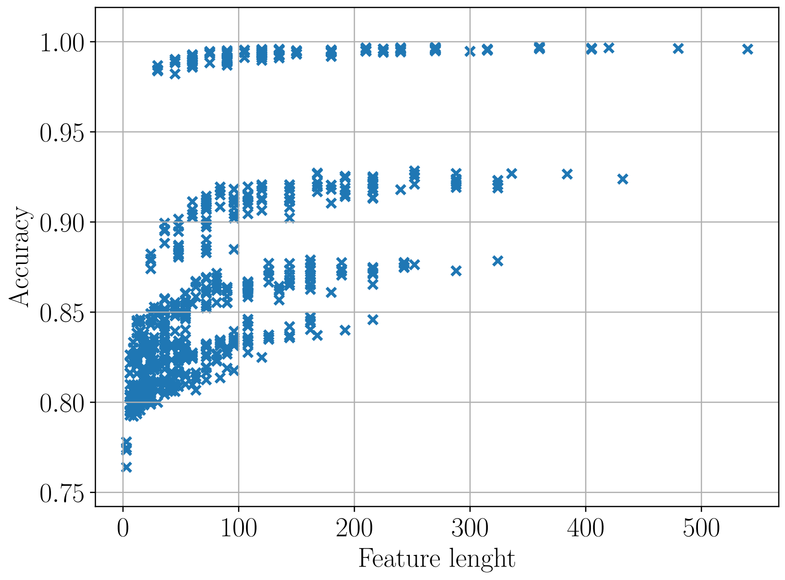


Figure 4.6: *Accuracy obtained with different feature vectors by changing the considered sets $\mathcal{B}$, $\mathcal{N}$ and $\mathcal{J}$. Each vector has a different length and provides a different accuracy result.*

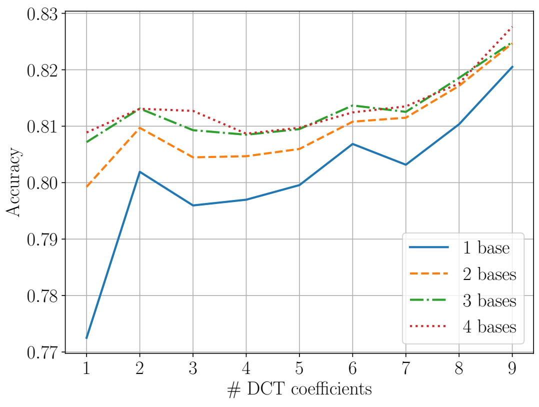
data-driven one based on Deep Learning; a solution based on hand-crafted features commonly used in the forensics literature.

Similarly to the solution in [110], we compared our approach with the Xception Convolutional Neural Network (CNN), as the first baseline method. According to the results in [110], this set of features seems to provide the best results over most of the considered datasets. Starting from the pre-trained model, we finetuned it on our dataset, following the same Leave-One-Group-Out strategy we adopted for the Random Forest training. We used 70% of the training data for the actual training, and the remaining 30% for validation, testing on the Leave-Out dataset. We resorted to Adam optimization algorithm, with an initial learning rate of 0.0001, training until reaching convergence on a validation plateau. We adopted the Keras implementation of Xception, performing the training in several hours on a workstation equipped with an NVIDIA Titan V Graphics Processing Unit (GPU), an Intel Xeon E5-2687W, and 256 GB of RAM.

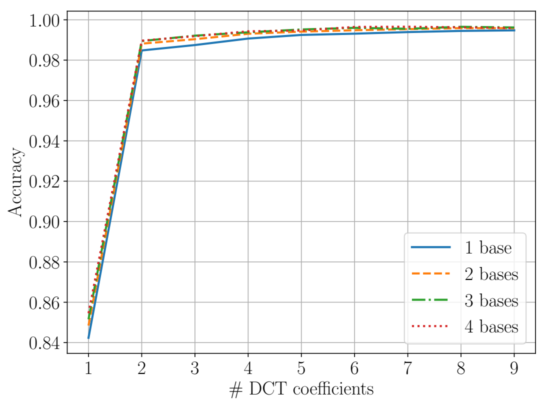
The second baseline method operates a linear Support Vector Machine (SVM) on a set of handcrafted steganalysis rich-features (as suggested in [110]). These features have been successfully used in image forgery detection tasks as well [153]. The model has been trained following the same train/test strategy used for Xception, using the Scikit Learn implementation of SVM.

Feature length and parameters

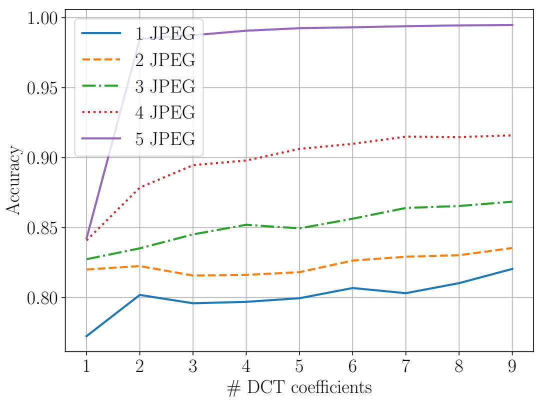
In the design of the proposed solution we considered different combinations of features obtained varying the parameters in the set $\mathcal{B}$, $\mathcal{N}$, $\mathcal{I}$ and changing feature vectors' lengths. As a matter of fact, it is necessary to evaluate how the vector length could impact on the classifier performance. Figure 4.6 shows the average test accuracy obtained on all datasets considering all possible 675 feature vectors. It is possible to notice that even the smallest feature vectors of just 3 elements enable achieving accuracy greater than 0.75. It is sufficient to use 50 features to have accuracy higher than 0.97.



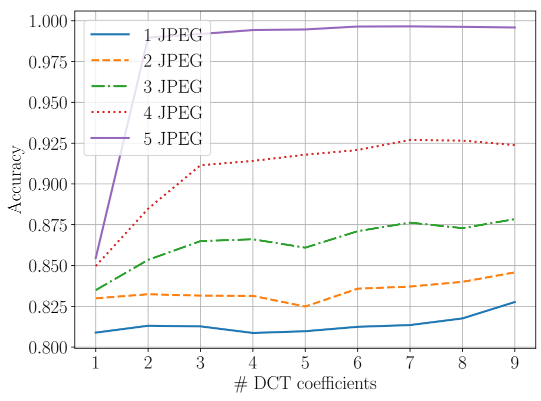
(a) *single Δ value*



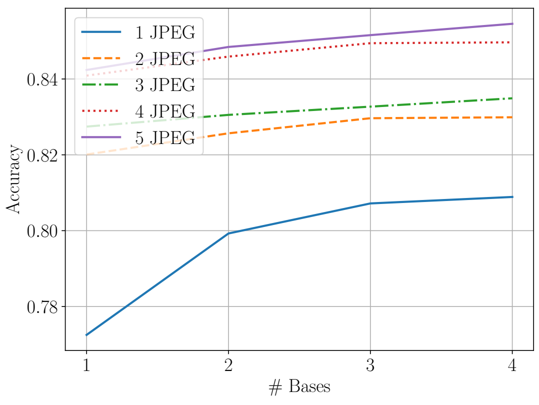
(b) *all Δ values*



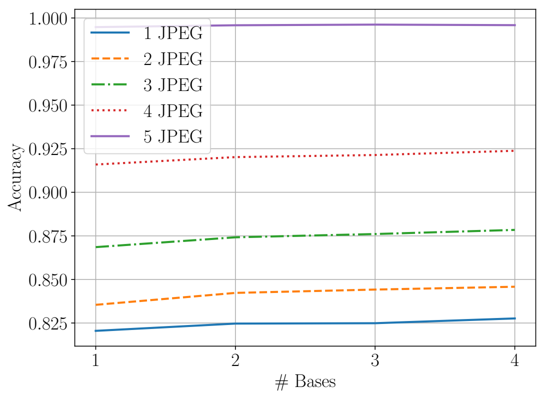
(c) *single b value*



(d) *all b value*



(e) *single n value*



(f) *all n value*

Figure 4.7: *Accuracy varying different parameters: (a) fix* 1 *JPEG compressions; (b) fix* 5 *JPEG compressions; (c) fix* 1 *base; (d) fix* 4 *bases; (e) fix* 1 *DCT coefficient; (f) fix* 9 *DCT coefficients. When we fix a single parameter, we average all results obtained by fixing that parameter to each possible value it can span.*

To gain a better insight on the effect of using different bases, DCT coefficients and quantization steps, we performed an analysis by keeping some parameters fixed, and just changing the others. Figure 4.7a and Figure 4.7b show the results with a single fixed quantization step value, and considering all the values, respectively. In both scenarios it is possible to notice that the greatest improvement is obtained when more than a single DCT coefficient is used. Moreover, the more the coefficients, the better the results. Figure 4.7c and Figure 4.7d report the results with features from a single fixed FD base and from all the considered bases, respectively. It seems that using more than one base only marginally improve the results. As a matter of fact, both figures are very similar. Finally, Figure 4.7e and Figure 4.7f display the accuracy values obtained from the features of a single DCT frequency and of the whole set of frequencies, respectively.

From these figures it is possible to note that, the more the considered quantization steps, the higher the accuracy for any other parameters set. This is not particularly surprising, as Benford's law is naturally linked to the used JPEG quantization.

4.3.4 Results

In this section we report all experimental results achieved to evaluate the proposed technique, including a comparison against baseline solutions. Finally, we provide some additional insights in terms of resilience to JPEG compression.

Comparison against baseline

In order to compare against the selected baselines solution, we finetuned Xception network and trained a linear SVM on the steganalysis features for each dataset according to the same procedure used for our Random Forest as suggested in [110].

Table 4.2 reports the breakdown of test accuracy scores for all datasets. The highest average accuracy among the considered methods is obtained by the proposed method, and it is higher than 0.99. It is also interesting to notice that the proposed solution is considerably better than the baseline CNN on `winter2summer`, `sats` and `lsun_bedroom`, which seem to be particularly though for the latter. These results highlight that, in order to properly train a very deep network like Xception, a much larger dataset probably is needed. However, this might be difficult to obtain in a reduced amount of time in a forensic scenario. On the contrary, the proposed feature vector is very compact, thus Random Forest does not suffer from a smaller training set. The baseline handcrafted method performs reasonably well, but the obtained accuracy is lower than that of the proposed method of almost 9% on average.

Resilience to JPEG compression

When images are shared online, JPEG compression is almost always applied in order to reduce network and storage requirements. Therefore, we measured the performance of the proposed method whenever a further JPEG compression is applied with different coding parameter configurations.

In a first scenario, GAN-generated and real images have been randomly JPEG compressed considering QFs distributed in $\{85, \ldots, 100\}$. The originally-trained detector (on non-compressed images) was then tested on this newly compressed dataset. In this situation, the proposed solution approaches a random guess accuracy. However, this situation is not completely unexpected. As a matter of fact, Benford's law is strictly tailored to JPEG compression. Therefore, scrambling with JPEG coefficients statistics through recompression has a high impact on Benford's features.

Table 4.2: *Accuracy results compared to baseline solutions for each dataset. Average accuracy (avg) is also reported. Best result per dataset in bold.*

Dataset	Proposed	Xception	Steganalysis
orange2apple	**98.13**	97.64	88.80
photo2ukiyoe	**100.00**	97.41	86.78
winter2summer	**100.00**	68.33	77.96
zebra2horse	**99.69**	89.58	91.01
photo2cezanne	**99.97**	95.91	95.88
photo2vangogh	**100.00**	93.75	94.68
photo2monet	**99.84**	94.08	94.80
facades	**100.00**	99.84	73.93
cityscapes	**100.00**	**100.00**	**100.00**
sats	**99.69**	73.00	90.92
lsun_bedroom	**100.00**	76.22	98.92
lsun_bridge	**99.89**	82.49	95.90
lsun_churchoutdoor	**99.99**	99.79	98.81
lsun_kitchen	**99.99**	87.26	99.49
lsun_tower	**99.98**	95.45	98.87
avg	**99.83**	89.64	91.03

We therefore considered a second scenario, which is more realistic as shown in [110]. If we know that images might be JPEG recompressed, we can also train our system on JPEG compressed images. We therefore re-trained our method and the baseline on compressed images, and tested them on compressed images. In this situation, results improve as expected. As a matter of fact, the proposed solution accuracy decreases, but still remains higher than 0.80. In particular, results depend on the specific datasets and GAN architecture. Indeed, all results related to ProGAN (i.e., last five datasets) show an almost optimal accuracy always higher than 0.99. Conversely, on Cycle-Gan images, only a couple of datasets exhibit accuracy grater than 0.70. In this situation, if computational cost is feasible for the adopted architecture, the baseline network might be preferable.

We then tested a third scenario, assuming that the analyst knows which is the QF adopted by the final JPEG compression stage (since it can be read from the bit stream). It is possible to train a different Random Forest classifier or Xception network for each QF. We therefore generated three versions of the dataset by recompressing it with QF 100, 95, and 90, respectively. For each QF, we trained the proposed method and Xception baseline using the aforementioned leave-one-group-out strategy. We did not consider steganalysis features anymore, as in [110] the authors already showed that they greatly suffer JPEG compression. Results are reported in Table 4.3. It is possible to notice that for high QFs, the proposed Benford-based method outperforms the baseline. Xception network shows better results starting from QF 90.

In the final testing scenario, we assume that the analyst wants to train a different classifier for each JPEG QF, and for each kind of image content. As an example, if the analyst is interested in detecting fake oranges with a given QFs, they might train only on the orange2apple dataset, rather than the others. In this situation (i.e., known QF and kind of GAN training dataset), both the proposed method and the Xception baseline achieve an almost perfect result for each QF (i.e., 100, 95 and 90).

Table 4.3: *Accuracy obtained using different JPEG QFs.*

QF	Dataset	Proposed	Xception
100	orange2apple	**94.50**	92.56
	photo2ukiyoe	**100.00**	98.50
	cityscapes	**100.00**	100.00
	lsun_tower	**100.00**	94.64
95	orange2apple	82.01	**90.66**
	photo2ukiyoe	97.00	**98.42**
	cityscapes	**99.99**	99.32
	lsun_tower	**99.80**	99.48
90	orange2apple	65.93	**85.61**
	photo2ukiyoe	92.01	**98.17**
	cityscapes	**100.00**	99.66
	lsun_tower	**99.60**	98.86

Analysis on faces

All results shown so far are obtained not considering GANs generating face images. This is due to two main reasons. First, GANs that were trained to generate face images produce particularly realistic results lately. This make face images harder to detect as GAN-generated compared to other kind of imagery. Indeed, shadows and lightning very often respect physics law, thus making Benford's law almost verified [138]. Second, face-generating GANs are often trained on common pristine face datasets [154], which makes the Leave-One-Group-Out testing strategy applied to now impracticable.

In the light of these considerations, we decided to create a specific dataset, composed by all the faces dataset in the original corpus, generated by ProGAN [152] (19 870 images), StarGAN [155] (50 000 images) and GlowGAN [156] (49 900 images), plus some additional images generated by the more recently proposed StyleGAN2 [109] (2000 images). As pristine faces, we always consider images from the Celeb-A dataset [154].

ProGAN is trained to generate realistic faces similar to those from Celeb-A dataset [154]. StarGAN and GlowGAN are trained to obtain faces with different characteristics (e.g., hair colors, smiles, etc.). Finally, StyleGAN2 produces images at different qualities depending on its configuration parameter $\psi = 0.5$ or $\psi = 1$ as suggested by the authors [109], starting from the Flickr-Faces-HQ Dataset [152]. Some random images from those generated by StyleGAN2 are shown in Figure 4.8. For each dataset, we train a Random Forest classifier considering 70% of the images as training set and 30% as test.

Table 4.4 shows the achieved results on each test set. It is possible to notice that StarGAN and GlowGAN seems to be easier to detect. On the contrary, ProGAN and StyleGAN2 looks more challenging. These promising preliminary results motivate some future work with more extended face image datasets, also comparing against other baselines and in presence of editing operations.

Figure 4.8: *Examples of faces generated with StyleGAN2 [109].*

Table 4.4: *Accuracy results for each face test dataset. Average accuracy (avg) is also reported. Accuracy higher than* 85% *are reported in bold.*

Dataset	**Proposed**
progan_celeba	79.75
stargan_black_hair	$\mathbf{97.26}$
stargan_blond_hair	$\mathbf{96.56}$
stargan_brown_hair	$\mathbf{96.76}$
stargan_male	$\mathbf{96.24}$
stargan_smiling	$\mathbf{96.06}$
glow_black_hair	$\mathbf{86.56}$
glow_blond_hair	$\mathbf{88.26}$
glow_brown_hair	$\mathbf{86.18}$
glow_male	$\mathbf{87.11}$
glow_smiling	83.04
stylegan2-0.5	77.18
stylegan2-1	72.63
avg	$\mathbf{87.96}$

4.4 Detecting Deepfake videos by model ensembling

In this section we propose a method for detecting Deepfake videos. We start from a well known state of the art network in the field of computer vision and we slightly modify it to add an attention mechanism. Then we put together different versions of the same network composing an ensemble that we use for the detection task.

4.4.1 Proposed method

We define the Deepfake video detection as a two-class classification problem. We work at frame level, instead of exploiting the temporal dimension of the video. Formally, given a frame image $\mathbf{I}$, we would like to build a model $M(\cdot)$ such that:

$$\hat{y} = M(\mathbf{I}), \tag{4.13}$$

where $\hat{y}$ represents the score provided by the model to classify the frame. Once obtained the scores for each frame in the video, we can proceed with a proper aggregation policy to obtain a score for the video. Our goal is to build the model $M(\cdot)$.

The proposed method is based on the concept of ensembling. Indeed, it is well-known that model ensembling may lead to better prediction performance with respect to a single model. We, therefore, focus on investigating whether and how it is possible to train different CNN-based classifiers to capture different high-level semantic information that complements one another, thus positively contributing to the ensemble for this specific problem.

To do so, we consider as starting point the EfficientNet family of models, proposed in [18] as a novel approach for the automatic scaling of CNNs. This set of architectures achieves better accuracy and efficiency with respect to other state-of-the-art CNNs and might be very useful to fulfil potential hardware and time constraints. Given an EfficientNet architecture, we propose to follow two paths to make the model beneficial for the ensembling. On the one hand, we propose to include an attention mechanism, which also provides the analyst with a method to infer which portion of the investigated video is more informative for the classification process. On the other hand, we investigate how siamese training strategies can be included in the learning process for extrapolating additional information about the data.

We provide more details about EfficientNet architecture with the proposed attention mechanism and the network training strategies in the following.

EfficientNet and attention mechanism

Among the family of EfficientNet models, we choose the EfficientNetB4 as the baseline for our work, motivated by the good trade-off offered by this architecture in terms of dimensions (i.e., number of parameters), run time (i.e., FLOPS cost) and classification performance. As reported in [18], with 19 millions of parameters and 4.2 billions of FLOPS, EfficientNetB4 reaches the 83.8% top-1 accuracy on the ImageNet [157] dataset. On the same dataset, XceptionNet, used as face manipulation detection baseline method by the authors of [145], reaches the 79% top-1 accuracy at the expense of 23 millions parameters and 8.4 billions FLOPS.

EfficientNetB4 architecture is represented within the blue block in Figure 4.9, where all layers are defined using the same nomenclature introduced in [18]. The input to the network is a squared color image $\mathbf{I}$, i.e., in our experiments, the face extracted from a video frame. As a matter of fact, authors of [145] recommend to track face information instead of using the full frame as input to the network for increasing the classification accuracy. Moreover, faces can be easily extracted from frames using any of the widely available face detectors proposed in the literature [158, 159]. The network output is a feature vector g of 1792 elements, defined as $\mathbf{g} = G(\mathbf{I})$. The final score related to the face is the result of a classification layer.

The proposed variant of the standard EfficientNetB4 architecture is inspired by the several contributions in the natural language processing and computer vision fields that make use of attention mechanisms. Works such as the transformer [160] and residual attention networks [161] show how it is possible for a neural network to learn which part of its input (being an image or a sequence of words) is more relevant for accomplishing the task at hand. In the context of video deepfake detection, it would be of great benefit to discover which portion of the input gave the network more information for its decision making process. We thus explicitly implement an attention mechanism similar to the one already exploited by the EfficientNet itself, as well as to the self-attention mechanisms presented in [162, 163]:

1. we select the feature maps extracted by the EfficientNetB4 up to a certain layer, chosen such that these features provide sufficient information on the input frame

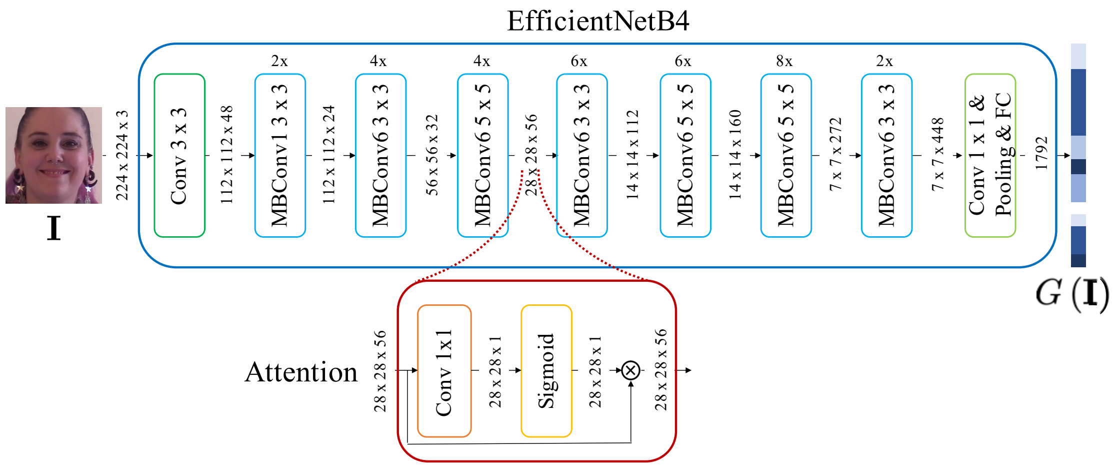


Figure 4.9: *Blue block: EfficientNetB4 model. If the red block is embedded into the network, an attention mechanism is included in the model, defining the proposed EfficientNetB4Att architecture.*

without being too detailed or, on the contrary, too unrefined. To this purpose, we select the output features at the third MBConv block which have size $28 \times 28 \times 56$;

2. we process the feature maps with a single convolutional layer with kernel size 1 followed by a Sigmoid activation function to obtain a single attention map;
3. we multiply the attention map for each of the feature maps at the selected layer.

For clarity's sake, the attention-based module is depicted in the red block of Figure 4.9.

On one hand, this simple mechanism enables the network to focus only on the most relevant portions of the feature maps, on the other hand it provides us with a deeper insight on which parts of the input the network assumes as the most informative. Indeed, the obtained attention map can be easily mapped to the input sample, highlighting which elements of it have been given more importance by the network. The result of the attention block is finally processed by the remaining layers of EfficientNetB4. The whole training procedure can be executed end-to-end, and we call the resulting network EfficientNetB4Att.

Network training

We train each model according to two different training paradigms: (i) end-to-end, and; (ii) siamese. The former represents a more classical training strategy. The latter aims at exploiting the generalization capabilities offered by the networks in order to obtain a feature descriptor that privileges the similarity between samples belonging to the same class. The ultimate goal is to learn a representation in the encoding space of the network's layers that well separates samples (i.e., faces) of the real and fake class.

End-to-end training. We feed the network with a sample face, and the network returns a face-related score $\hat{y}$. Notice that this score is not passed through a Sigmoid activation function yet. The weights update is led by the commonly used LogLoss function:

$$\mathscr{L}_L = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log\left(S(\hat{y}_i)\right) + (1-y_i)\log\left(1-S(\hat{y}_i)\right)\right], \tag{4.14}$$

where $\hat{y}_i$ represents the i-th face score, $y_i \in \{0, 1\}$ the related face label. Specifically, label 0 is associated with faces coming from real pristine videos and label 1 with fake videos. N is the total number of faces used for training and $S(\cdot)$ is the Sigmoid function.

Siamese training. Inspired by computer vision works that generate local feature descriptors using CNNs, we adopt the triplet margin loss, first proposed in [164].

Recalling that $G(\mathbf{I})$ is the non-linear encoding obtained by the network for an input face $\mathbf{I}$ (see Figure 4.9), being $\|\cdot\|_2$ the L_2 norm, the triplet margin loss is defined as

$$\mathscr{L}_T = \max(0, \mu + \delta_+ - \delta_-), \tag{4.15}$$

with $\delta_+ = \|G(\mathbf{I}_a) - G(\mathbf{I}_p)\|_2$, $\delta_- = \|G(\mathbf{I}_a) - G(\mathbf{I}_n)\|_2$ and μ is a strictly positive margin. In this case $\mathbf{I}_a$, $\mathbf{I}_p$ and $\mathbf{I}_n$ are, respectively:

- $\mathbf{I}_a$ the *anchor* sample (i.e., a real face);
- $\mathbf{I}_p$ a *positive* sample, belonging to the same class as $\mathbf{I}_a$ (i.e., another real face);
- $\mathbf{I}_n$ a *negative* sample, belonging to a different class than $\mathbf{I}_a$ (i.e., a fake face).

We then finalize the training by finetuning a simple classification layer on top of the network, following the end-to-end approach described before.

4.4.2 Experiments

In this section we present the experimental setup we adopted to validate the proposed method. First we describe the considered datasets, then we describe the network architectures and the training procedure we designed for them.

Dataset

We test the proposed method on two different datasets: FF++ [145]; DFDC [146].

FF++ is a large-scale facial manipulation dataset generated using automated state-of-the-art video editing methods. In detail, two classical computer graphics approaches are used, i.e., Face2Face [117] and FaceSwap [17], together with two learning-based strategies, i.e., DeepFakes [119] and NeuralTextures [118]. Every method is applied to 1000 high quality pristine videos downloaded from YouTube, manually selected to present nearly front-facing subjects without occlusions. All the sequences contain at least 280 frames. Eventually, a database of more than 1.8 million images from 4000 manipulated videos is built. In order to simulate a realistic setting, videos are compressed using the H.264 codec. High quality as well as low quality videos are generated using a constant rate quantization parameter equal to 23 and 40, respectively.

DFDC is the training dataset released for the homologous Kaggle challenge [146]. It is composed by more than $119\,000$ video sequences, created specifically for this challenge, representing both real and fake videos. The real videos are sequences of actors taking into account diversity in several axes (gender, skin-tone, age, etc.) recorded with arbitrary backgrounds to bring visual variability. The fake videos are created starting from the real ones and applying different DeepFake techniques, e.g., different face swap algorithms. Notice that we do not know the precise algorithms used to generate fake videos, since for the time being the complete dataset (i.e., with the public and private testing sequences and possibly an explanation of the creation procedure) has not been released yet. The sequence length for each video is approximately around 300 frames, and the classes are strongly unbalanced towards the fake one, counting roughly 100 000 fakes and 19 000 reals.

Networks

In our experiments, we consider the following networks:

- XceptionNet, since it is the best performing model used in [145], thus being the natural yardstick for our experimental campaign;
- EfficentNetB4, as it achieves better accuracy and efficiency than other existing methods [18];
- EfficentNetB4Att, which should discriminate relevant parts of the face sample from irrelevant ones.

Each model is trained and tested separately over both the considered datasets. Specifically, regarding FF++, we consider only videos generated with constant rate quantization equal to 23.

XceptionNet is trained using the same approach of [145], whereas the two EfficientNet models are trained following the end-to-end as well as the siamese fashion described in Section 4.4.1. In doing so, we end up with 4 trained models: EfficientNetB4 and EfficientNetB4Att which are trained with the classical end-to-end approach, together with EfficientNetB4ST and EfficientNetB4AttST, trained using the siamese strategy. All these EfficientNetB4-derived models can contribute to the final ensembling.

Setup

We adopt a different split policy for each dataset. We split DFDC according to its folder structure, using the first 35 folders for training, folders from 36 to 40 for validation and the last 10 folders for testing. Regarding FF++, we use a similar split as in [145] selecting 720 videos for training, 140 for validation and 140 for test from the pool of original sequences taken from YouTube. The corresponding fake videos are assigned to the same split. All the results are shown on the test sets.

In our experiments, we only consider a limited number of frames for each video. In training phase, this choice is motivated by two main considerations: (i) when using a really small amount of frames per video, there is a strong tendency to overfit; (ii) increasing the number of frames does not improve performances in a justifiable manner. This phenomenon can be noticed in Figure 4.10, which reports training and validation losses as a function of training iterations, selecting a variable amount of frames per video. It is worth noting that the minimum validation loss does not improve selecting 15 frames per video instead of 32, however choosing 32 frames per video helps to prevent overfitting. Figure 4.10, there is a strong tendency to overfit when using a small number of frames. With this in mind, we limit the number of analyzed frames from each sequence to 32 for both training and testing phases. Even in this setting, the

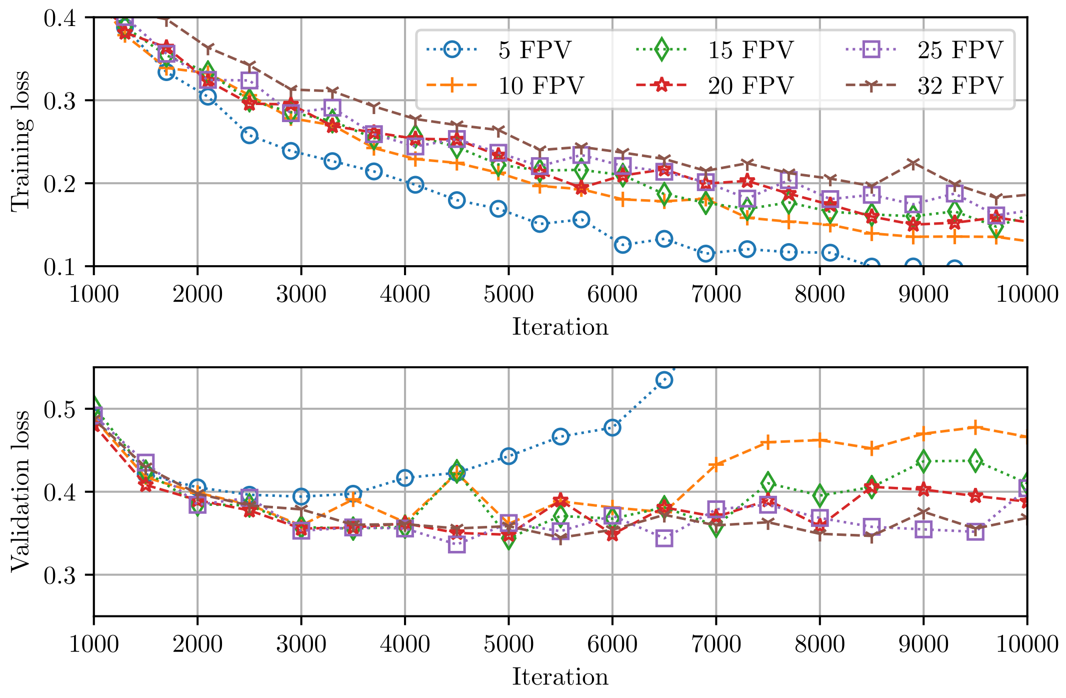


Figure 4.10: *Training and validation loss curves for XceptionNet on FF++, while varying the number of frames per video (FPV).*

dimensions of the datasets remain remarkable: for the FF++, we end up with roughly 1.6 million images, while for the DFDC with 3.4 million frames.

In this perspective, we can further reduce the amount of data processed by the networks by recalling that not all the frame information is useful for the deepfake detection process [145]. Indeed, we can mainly focus our analysis on the region where the face of the subject is located. Consequently, as a pre-processing step, we extract from each frame the faces of the scene subjects using the BlazeFace extractor [159], that, in our experiments, proved to be faster than the MTCNN detector [158] used by the authors of [145]. In case more than one face is detected, we keep the face with the best confidence score. The resulting input for the networks is the squared color image $\mathbf{I}$ introduced in Section 4.4.1, of size 224×224 pixel.

During training and validation, to make our models more robust, we perform data augmentation operations on the input faces. In particular, we randomly apply downscaling, horizontal flipping, random brightness contrast, hue saturation, noise addition and finally JPEG compression. Specifically, we resort to Albumentation [165] as our data-augmentation library, while we use Pytorch [166] as Deep Learning framework. We train the models using Adam [167] optimizer with hyperparameters equal to $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, and initial learning rate equal to 10^{-5}.

Independently from the training strategy used, given the size of the datasets, we never train our networks for a complete epoch. Specifically:

- for the *end-to-end training*, we either train for a maximum of 20k iterations, indicating as iteration the processing of a batch of 32 faces (16 real, 16 fake) taken

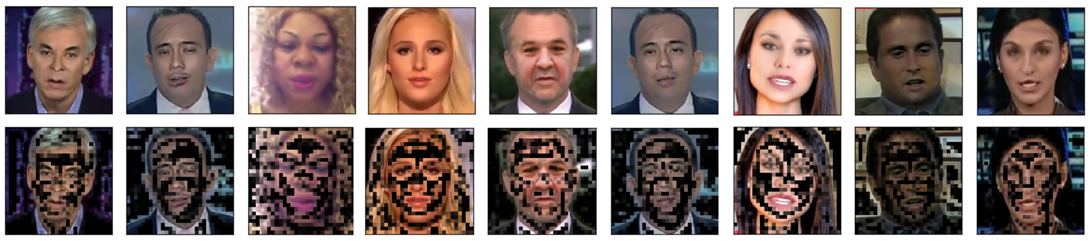


Figure 4.11: *Effect of the attention on faces under analysis. Given some faces to analyze (top row), the attention network tends to select regions like eyes, mouth and nose (bottom row). Faces have been extracted from FF++ dataset.*

randomly and evenly across all the videos of the train split, or until reaching a plateau on the validation loss. Validation of the model in this context is performed every 500 training iterations, on 6000 samples taken again evenly and randomly across all videos of the validation set. The initial learning rate is reduced of a 0.1 factor if the validation loss does not decrease after 10 validation routines (5000 training iterations), and the training is stopped when we reach a minimum learning rate of 1×10^{-10};

- for the *siamese training*, the feature extractor is trained using the same number of iterations, validation routine and learning rate scheduling of the end-to-end training. The main difference lies in the different loss function used (as explained in Section 4.4.1), and in the composition of the batch, which in this case is made by 12 triplets of samples (6 real-fake-fake, 6 fake-fake-real) selected across all videos of the set considered. Regarding the parameter μ in Equation (4.15), we set it to 1 after some preliminary experiments. The fine-tuning of the classification layer is then executed in a successive step following the end-to-end training paradigm with the hyperparameters specified above.

We finally run our experiments on a machine equipped with an Intel Xeon E5-2687W-v4 and a NVIDIA Titan V. The code to replicate our tests is freely available [1].

4.4.3 Results

EfficientNetB4Att explainability

In order to show the effectiveness of the attention mechanism in extracting the most informative content of faces, we evaluate the attention map computed on a few faces of FF++. Referring to Figure 4.9, we select the output of the Sigmoid layer in the attention block, which is a 2D map with size 28×28. Then, we up-scale it to the input face size (224×224), and superimpose this to the input face. Results are reported in Figure 4.11.

It is worth noting that this simple attention mechanism enables to highlight the most detailed portion of faces, e.g., eyes, mouth, nose and ears. On the contrary, flat regions (where gradients are small) are not informative for the network. As a matter of fact, it has been shown several times that artefacts of deepfake generation methods are mostly localized around facial features [16]. For instance, roughly modeled eyes and teeth, showing excessively white regions, are still the main trademarks of these methods.

[1] `https://github.com/polimi-ispl/icpr2020dfdc`

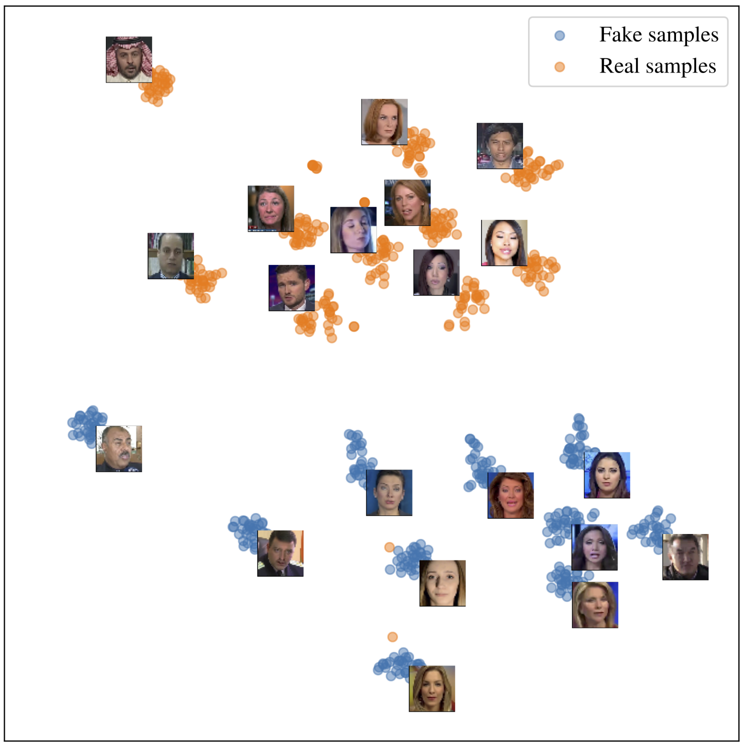


Figure 4.12: *t-SNE visualization of features obtained by EfficientNetB4AttST with siamese training. Faces have been extracted from FF++ dataset.*

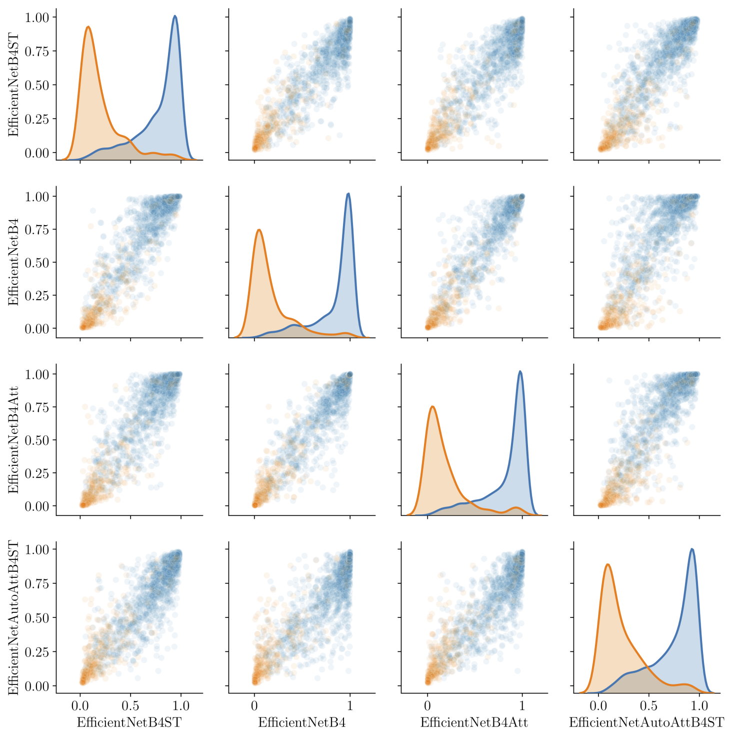
Siamese features

In order to understand whether the features produced by the encoding of the network when trained in siamese fashion are discriminatory for the task, we computed a projection over a reduced space using the well known algorithm t-SNE [168]. In Figure 4.12 we show the projection obtained by means of EfficientNetB4AttST starting from 20 FF++ videos.

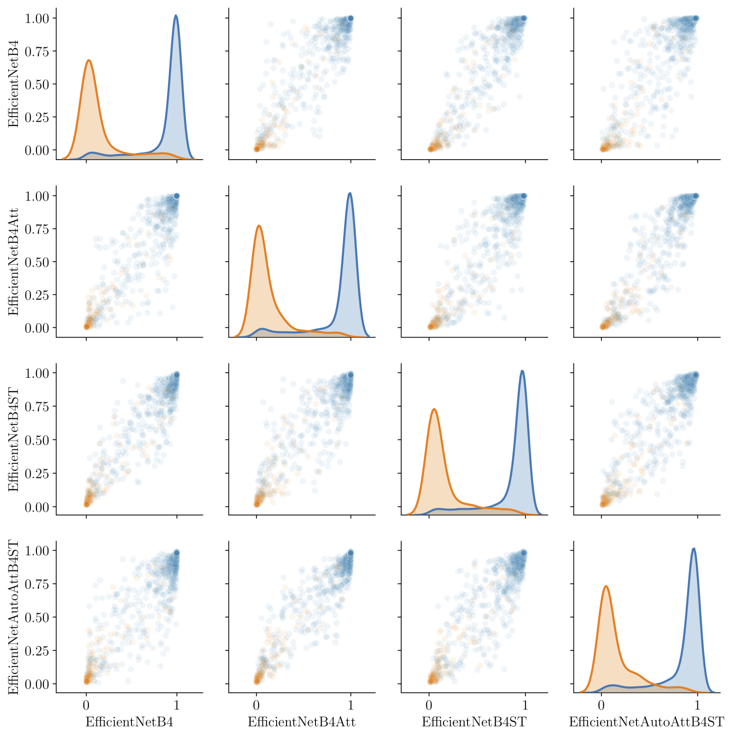
We can clearly see how frames of the same videos clusters into small sub-regions. More importantly, all the real samples cluster into the top region of the chart, whereas the fake samples are in the bottom region. Frames of the same videos clusters into smaller sub-regions. This justifies the choice to adopt this particular training paradigm in addition to the classical end-to-end approach.

Architecture independence

As we want to understand whether the different networks can be used in an ensemble, we explore whether the scores extracted by each model are independent to some extent. In Figure 4.13, all plots outside of the main diagonal show that different networks provide slightly different scores for each frame. Indeed, the point clouds do not perfectly align on a shape that can be easily described by a simple relation. This motivates us in using the different trained models in an ensemble way. If all networks were perfectly correlated, this would not be reasonable.



(a) *FF++*



(b) *DFDC*

Figure 4.13: *Pair-plot showing the score distribution for real (orange ●) and fake (blue ●) samples for each pair of networks on FF++ (a) and DFDC (b) datasets.*

Table 4.5: *Area Under the Curve (AUC) and LogLoss obtained with different network combinations over all the datasets. Top-3 results per column in bold, baseline in italics.*

Xception	EfficientNet				AUC		LogLoss	
Net	B4	B4ST	B4Att	B4AttST	FF++	DFDC	FF++	DFDC
✓					*0.9273*	*0.8784*	*0.3844*	*0.4897*
	✓				0.9382	0.8766	0.3777	0.4819
		✓			0.9337	0.8658	0.3439	0.5075
			✓		0.9360	0.8642	0.3873	0.5133
				✓	0.9293	0.8360	0.3597	0.5507
	✓	✓			0.9413	**0.8800**	0.3411	0.4687
	✓		✓		0.9428	**0.8785**	0.3566	0.4731
	✓			✓	0.9421	0.8729	0.3370	0.4739
		✓	✓		0.9423	0.8760	0.3371	0.4770
		✓		✓	0.9393	0.8642	**0.3289**	0.4977
			✓	✓	0.9390	0.8625	0.3515	0.4997
	✓	✓	✓		**0.9441**	**0.8813**	0.3371	**0.4640**
	✓	✓		✓	0.9432	0.8769	**0.3269**	**0.4684**
	✓		✓	✓	**0.9433**	0.8751	0.3399	0.4717
		✓	✓	✓	0.9426	0.8719	0.3304	0.4800
	✓	✓	✓	✓	**0.9444**	0.8782	**0.3294**	**0.4658**

Face manipulation detection capability

In this section, we report the average results achieved by the baseline network (i.e., XceptionNet) and the 4 proposed models (i.e., EfficientNetB4, EfficientNetB4Att, EfficientNetB4ST and EfficientNetB4AttST). We also verify our guess behind the use of an ensemble, specifically combining two, three or even all the proposed models. In this case, the final score associated with a face is simply computed as the average between the scores returned by the single models.

In Table 4.5 we report the AUC (computed binarizing the network output with different thresholds) and LogLoss obtained in our experiments. Results are provided in a *per-frame* fashion. Analyzing these results, it is worth noting that the strategy of model ensembling generally awards in terms of performances. As somehow expected, best top-3 results are always reached by a combination of 2 or more networks, meaning that network fusion helps both the accuracy of the deepfake detection (estimated by means of AUC) and the quality of the detection (estimated by means of LogLoss measure). Indeed, on both datasets, LogLoss and AUC are always better than the baseline.

Kaggle results

To gain a deeper insight on the proposed solution performance, we also participated to the DFDC challenge on Kaggle [146] as *ISPL* team. The ultimate goal of the competition was to build a system able to tell whether a video is real or fake. The DFDC dataset used here represents the training dataset released by the competition host, while the evaluation is performed over two different testing datasets: (i) the public test dataset; (ii) the private test dataset. Participants were not aware of the composition of those datasets (e.g., the provenance of the sequences, the techniques used for generating fakes, etc.), apart from the number of videos in public test set, which is roughly 4000. The final solution proposed by our team was an ensemble of the 4 proposed models, which led us to top 2% on the leaderboard computed against the private test set.

4.5 Conclusions

In this chapter we studied in deep the problem of generated images and videos, considering both the model based statistical properties of the generation process and the data driven model ensembling. In the first case, we proposed a study on the use of the well-known Benford's law for the task of GAN-generated image detection, by defining a strategy to extract Benford-related features from an image relying on different divergence definitions. We also showed how to combine these features in order to better exploit different bases as well as DCT frequencies. Using these features, we performed a series of experiment based on a simple Random Forest classifier in order to study the amount of information captured by the features, rather than focusing on specializing a complex classifier. Results show that GAN-generated images often fail in respecting Benford's law, thus can be discriminated from natural pictures. However, some kind of CNN architectures seem to produce images that are harder to detect than others.

In addition to face image manipulation, we tackled the detection of facial manipulation in video sequences, targeting classical computer graphics as well as Deep Learning generated fake videos. The proposed method takes inspiration from the family of EfficientNet models and improves upon a recently proposed solution, investigating an ensemble of models trained using two main concepts: (i) an attention mechanism which generates a human comprehensible inference of the model, increasing the learning capability of the network at the same time; (ii) a triplet siamese training strategy which extracts deep features from data to achieve better classification performances. Results evaluated over two publicly available datasets containing almost $120\,000$ videos reveal the proposed ensemble strategy as a valid solution for the goal of facial manipulation detection.

Part II

Physical Integrity

CHAPTER 5

Physical integrity with X-ray imaging

Assessing *physical integrity* of products is a fundamental part of quality control in industry. This can be done with several different techniques. The most interesting ones are surely non-destructive methods, i.e., solutions that enable to inspect the object under analysis not altering its status. For instance, if we think about food safety, the goal of *physical integrity* checks is to ensure that food has not been contaminated, avoiding opening all food containers under analysis with the risk of introducing additional contamination. Among these methods, Hyperspectral X-ray analysis is a very powerful and flexible technique to inspect the inside of the products to find imperfections or contaminants.

In this chapter we introduce the topic of Hyperspectral X-ray analysis aimed at food safety. This is just one of the possible application of this kind of analysis, yet we consider it as one of the most interesting to investigate due to its diffusion and effectiveness in industrial pipelines [169]. This chapter serves as the foundation for the next two chapters, as it provides the reader with a few elemental concepts that will be used later on, i.e., the considered X-ray acquisition system, and a physical law that links X-ray acquisitions to the physical object under analysis. In the next two chapters we will investigate more in detail two aspects of Hyperspectral X-rays analysis: the denoising of X-ray acquisitions, and the classification of plastic polymers from the acquired data.

This introductory chapter is organized as follows: Section 5.1 provides some state of the art for the use of X-ray analysis in food safety. Section 5.2 describes the X-ray acquisition system we consider to obtain the hyperspectral acquisitions used in the coming chapters. Section 5.3 provides some background on the Lambert-Beer law, a physical law we exploit in the next chapters. Finally, Section 5.4 concludes the chapter.

5.1 Physical integrity in the food industry

Industrial food is a broad presence in alimentation nowadays. More and more groceries are prepared, cooked and packed in industrial pipelines, with a possible risk of contamination from foreign bodies, which can be harmful if ingested. For this reason, authorities like the European Commission have developed a series of safety guidelines inspired by the Hazard Analysis and Critical Control Points (HACCP) [6]. Producers are forced to respect these standard rules to lower the risk of contamination at any point in the production chain. Unfortunately, this does not ensure complete safety by itself: the food still needs to be inspected.

Among the possible ways of analysing the food, we can find intrusive tests which require access to the inspected food. Biological, physical and chemical analysis are often part of this category [170]. Non-intrusive tests are also very common [171], and they have the advantage of not requiring physical access to the food, proving particularly suited for post-packaging analysis. This category includes all the various types of scans performed on the package while transporting it on a conveyor belt. X-ray scans are commonly adopted in industry for finding contaminants with a high density with respect to the food (typically metals), which appear as darker areas in the acquisition [7, 172–174]. One of the shortcomings of the traditional X-ray approaches is that they are not suited for spotting low-density contaminants (i.e., certain plastic polymers, organic material, wood), which typically do not block enough radiation to be visible in the output image [175–179]. With hyperspectral X-rays, we can divide the radiation energy into sub-bands to be analyzed separately. This brings more information in the output image and allows for detecting even low-density contaminants [28, 29].

X-ray penetration depends on a series of factors: the atomic number, density and thickness of the elements composing the object under analysis [180], [181]. Therefore, X-ray hyperspectral images can be used as signatures of different materials. Additionally, if an object is a composite of different elements, it is possible to linearly combine the absorption effect of the different components [182]. If we know the kind of food we are analysing and we want to detect physical contaminants within it, it is possible to subtract the specific food absorption signature from the performed X-ray acquisition, thus leaving only the possible contaminant absorption signature within the measurements [183]. This enables developing classification techniques that are agnostic to the kind of food a contaminant may be corrupting. In other words, recognizing a contaminant alone helps us recognizing the contaminant buried in food.

5.2 Acquisition system

With reference to Figure 5.1, the hyperspectral image acquisition system considered in this work is composed by a X-ray generator, a conveyor transporting the objects under analysis (e.g., food containers) and a hyperspectral X-ray detector. This is a very common system architecture for industrial inspection. The source of the X radiation is an X-ray tube, i.e., a vacuum tube containing a cathode and an anode. X-rays are generated by directing a stream of high speed electrons from the cathode to the anode. The detector is a linear sensor that measures the intensities (at different spectral frequencies, commonly named X-ray energies due to the Planck-Einsten Relation [184]) of photons that have not been absorbed by objects in front of it, i.e., the photons that pass through

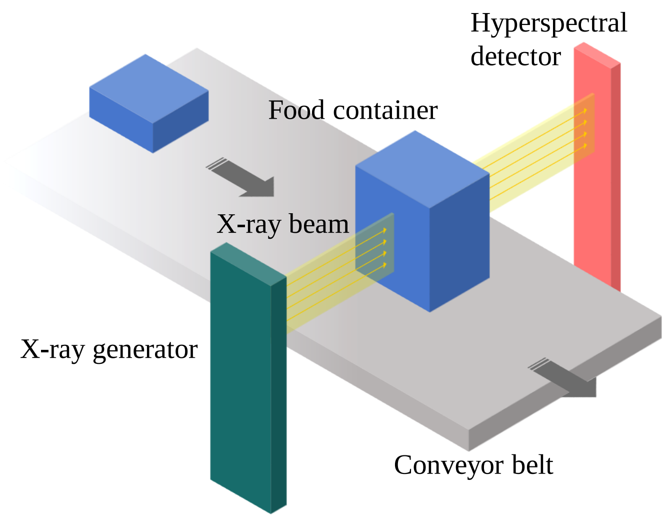


Figure 5.1: *Representation of the X-ray acquisition system for food contaminant classification.*

the object reaching the sensor.

The acquired spectrum is divided into B photons energy intervals, also called energy bins. Therefore, each pixel of the acquired image is a vector of size $1 \times B$ containing the intensities at all the energy bins. Since the detector is a push broom sensor of H vertically aligned pixels, the acquired 2D signal is a linear hyperspectral image $\mathbf{I}$ (called image in the rest of the chapter) with the size of $H \times B$ samples, and each value $[\mathbf{I}]_{ij},\ i = \{1, \dots, H\},\ j = \{1, \dots, B\}$ represents the intensity of received photons for a given spatial position and a given energy bin. Figure 5.2 shows an example of the sensor's output, while Figure 5.3 shows an example of the 1D signal we can obtain by selecting a particular pixel from the 2D acquisition. We will use both kinds of signal in the next chapters.

It is important to point out that, in normal operating condition of the system, the scanned object is moving on the conveyor belt. It is therefore paramount to develop timely and efficient solutions for data denoising and processing in order to work in real-time environments.

5.3 Lambert-Beer law

Lambert-Beer law relates the absorption of light to the material through which the light passes. Similarly, the absorption of X-rays at each energy E is related to the material through which the beam passes by the following equation:

$$\lambda_{\text{OUT}}(E) = e^{-\mu(E)\Delta x}\lambda_{\text{IN}}(E), \tag{5.1}$$

where $\lambda_{\text{IN}}(E)$ is the average number of incident X-ray photons with energy E, $\lambda_{\text{OUT}}(E)$ is the average number of transmitted ones, $\mu(E)$ is the linear attenuation coefficient of the material and Δx is the thickness of material through which X-rays have travelled. When a X-ray beam is acquired and digitalised by a sensor, energies E are discretised

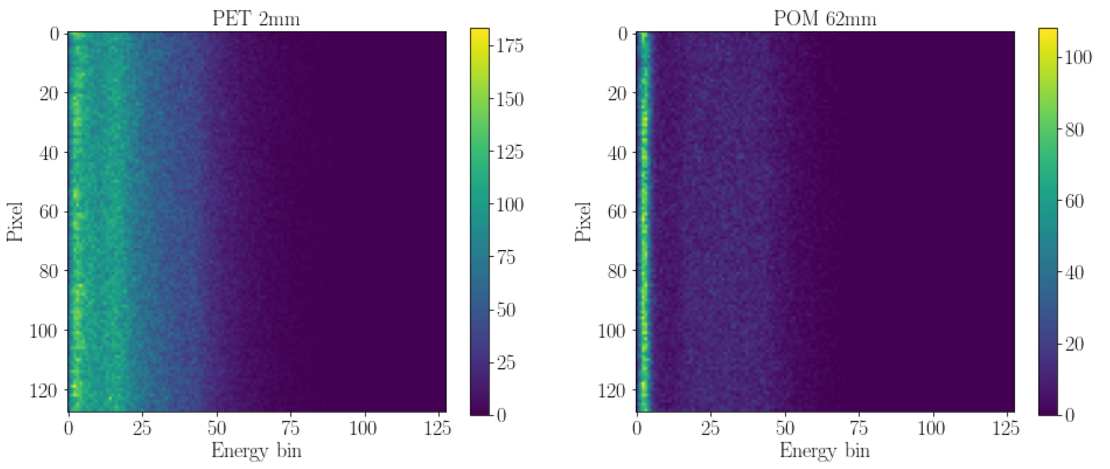


Figure 5.2: *Example of acquired* 2*D Hyperspectral X-ray signals. The energy range has been divided into* $B = 128$ *energy bins.*

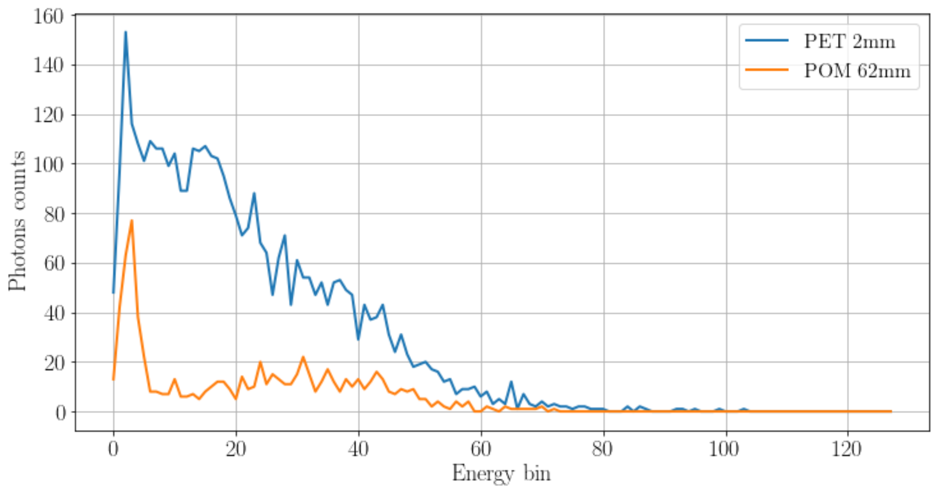


Figure 5.3: *Examples of* 1*D signals obtained by selecting a particular pixel from* 2*D acquisitions. The energy range has been divided into* $B = 128$ *energy bins.*

into B energy bins. As Lambert-Beer law holds for every energy bin, we can rewrite it in vectorial form as:

$$\boldsymbol{\lambda}_{\text{OUT}} = \text{diag}[e^{-\mu_b \Delta x}]\boldsymbol{\lambda}_{\text{IN}}, \tag{5.2}$$

where μ_b is the linear attenuation coefficient for the b-th energy bin, $\text{diag}\,[\cdot]$ is a $B \times B$ diagonal matrix, and both $\boldsymbol{\lambda}_{\text{OUT}}$ and $\boldsymbol{\lambda}_{\text{IN}}$ are B-element vectors. With this notation at hand, the sensor reading $\boldsymbol{y}$ is a measure of $\boldsymbol{\lambda}_{\text{OUT}}$.

In principle, we could exploit this relation to bind the absorption of X-rays at each energy bin with two physical parameters: (i) the effective atomic number Z_{eff} of the material and (ii) the density-width $\rho\Delta x$ [19,20,182,185]. In practice, we experimented that we cannot use the Lambert-Beer law directly on the measured mean acquisition, due to a series of distortions which invalidate the law. Still, we would like to exploit the fact that once we fix the X-ray tube and sensor's parameters, we can completely determine the shape of the acquired spectrum by only two parameters: Z_{eff} and $\rho\Delta x$.

5.4 Conclusions

This chapter introduced the reader to the problem of *physical integrity* in the food industry. We have seen why non-intrusive techniques may be useful to inspect the inside of objects under analysis in a non-destructive way, and how X-ray acquisition are considered a possible solution in this field.

Given the impact of X-ray-based methods in the *physical integrity* world, we described the specific acquisition system we have decided to use in our work. This system involves a conveyor belt that moves objects in between an X-ray source and a receiver, as it is typically done in common X-ray analysis. Finally, we quickly introduce the principle behind Lambert-Beer law, which will be exploited in the coming chapters.

The next chapter builds upon this one, by reporting our work on denoising of the acquired X-ray data.

CHAPTER 6

Hyperspectral X-ray Denoising Techniques

Due to their nature, Hyperspectral X-ray acquisitions are subject to a significant amount of noise. The main reason for this noise is the stochasticity of the photon emission process [186]. Therefore, Hyperspectral X-ray data need to be adequately preprocessed before their usage to ensure good quality *physical integrity* analysis results.

Referencing the system depicted in Figure 5.1, we know that the system's noise follows a Poissonian distribution independent in each energy sub-band of the spectrum [187]. The simplest way to obtain a noise-free acquisition is to average together a certain number of noisy acquisitions. Unfortunately, this averaging operation is often too slow for the typical applications of food safety, which often require processing to happen in real-time. This forces us to find alternative denoising procedures. As explained in Section 5.3, given a particular material, the shape of its ideal acquired signal is, in principle, determined by a couple of physical parameters. We want to exploit this information while developing an effective denoising algorithm.

This chapter treats the aspect of denoising Hyperspectral X-ray acquisitions, comparing model-based and data-driven solutions for this task. We first propose a comparison between Wiener filter [188] and the popular AutoEncoder (AE) neural network to assess if the denoising could be effectively done in a data-driven fashion. The rationale is that, if proven possible, a better denoising neural network could be developed, exploiting the abundance of data we can collect with the acquisition system. As a second contribution, we take into account different versions of AEs, and we compare them for the denoising task under strict conditions. We find particularly suited the AEs neural network family for their ability to find an encoding of the input signal of arbitrary dimensions. By forcing the input to be reconstructed with just a few parameters (e.g., the network latent space), the network should learn to keep only the relevant information and remove the noise. Driven by the Lambert-Beer law (Section 5.3), which describes a parameterization of each material using just two parameters, we focus on a 2-dimensional latent space for the AEs.

The chapter is organized as follows: Section 6.1 gives important details on the concepts of Wiener filter and AE that we use throughout the chapter. Section 6.2 describes the proposed method on the comparison between a model-based and a data-driven technique for denoising. Section 6.3 describes the proposed method on the use of AE variants for the denoising task. Finally, Section 6.4 draws common conclusions between the two proposed methods and concludes the chapter.

6.1 Background

In this section we introduce some background on Wiener filtering and AutoEncoders (AEs), as these are the main tools used in the rest of the chapter.

6.1.1 Wiener filter for image restoration

Wiener filtering is commonly adopted in image restoration problems [189–192], since it performs a statistical estimation of an unknown signal while taking noise into account. Considering a noise model as the following:

$$\mathbf{I} = \mathbf{I}_0 + \mathbf{N}, \tag{6.1}$$

in which $\mathbf{I}_0$ is the clean image and $\mathbf{N}$ an additive noise, Wiener filter approach estimates $\hat{\mathbf{I}}$ such that the Mean Squared Error (MSE) between $\mathbf{I}_0$ and $\hat{\mathbf{I}}$ is minimized. $\hat{\mathbf{I}}$ is defined as it follows:

$$[\hat{\mathbf{I}}]_{ij} = {F_2}^{-1}\left([F_2(\mathbf{I})]_{ij}\,\frac{[\mathbf{S}_I]_{ij}}{[\mathbf{S}_{I_0}]_{ij} + [\mathbf{S}_N]_{ij}}\right), \tag{6.2}$$

where $F_2(\cdot)$ is the 2D Fourier transform, $\mathbf{S}_N = |F_2(\mathbf{N})|^2$ represents the power spectrum of the noise, and $\mathbf{S}_I = |F_2(\mathbf{I})|^2$ is the power spectrum of the clean image. This method is very effective provided that $\mathbf{S}_N$ and $\mathbf{S}_I$ are estimated in a quite accurate way and model's assumptions hold [188].

6.1.2 AutoEncoders

An AutoEncoder (AE) is a specific kind of neural network. The purpose of such a network architecture is to learn a low dimensionality representation of an image (latent representation), and then reconstruct the input from it [193]. The rationale behind this technique is to build a representation of the input such that the reconstructed output is derived from its most robust features. A particular kind of AE is the Convolutional AutoEncoder (CAE) (Figure 6.1). It is composed by an encoding part, in which Convolution and Pooling layers are employed to reduce the input dimensionality, and a decoding part, in which Deconvolution (i.e. Transposed Convolution) and Upsampling layers are employed to expand the latent representation dimensionality up to the input shape. We can define the CAE as a function $C(\boldsymbol{\theta}, \cdot)$ such that:

$$\hat{\mathbf{I}} = C(\boldsymbol{\theta}, \mathbf{I}), \tag{6.3}$$

where $\mathbf{I}$ is the input image, $\hat{\mathbf{I}}$ is the output image, matching the shape of $\mathbf{I}$ and $\boldsymbol{\theta}$ is the AE weights' vector that must be learned with a suitable training procedure.

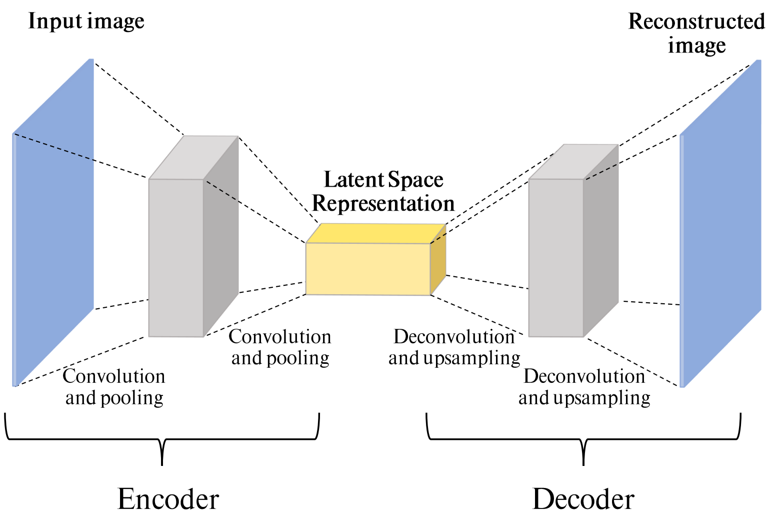


Figure 6.1: *Typical Convolutional AutoEncoder structure. An input image is mapped to its latent representation by the encoder. The decoder turns the latent representation into an estimate of the input data.*

AEs are one of the most used neural networks in literature when it comes to denoising. A method for denoising medical images with an eye on training sample's size is proposed in [194]. Also 1D signals are commonly processed with AE. An example is [195] where ECG signals are processed to remove various acquisition artefacts. In the last years, Variational AutoEncoder (VAE) was introduced as a generic inference framework for probabilistic models [196] but has been more recently adopted as a denoising network. Radar signals are cleaned from disturbs with a VAE in [197], while radio tomographic imagery is tackled in [198]. When using VAEs as a predictive model, we must take into account its probabilistic nature, which can lead to undesired stochasticity in the output. One recent work [199] shows how we can slightly change the perspective of seeing a VAE to keeping most of its advantages while keeping it deterministic. We can consider the random process of sampling the latent space (as proposed by the original VAE paper) as a random noise injection in the latent space of a deterministic AE. In this way, the injected noise augments the input data. The decoder has to keep this into account in learning the most appropriate input reconstruction, with the practical effect of making the network more robust. We call this particular network paradigm Augmented AutoEncoder (AuAE), and we refer to further sections for a more detailed explanation of this phenomenon.

6.2 Data driven and model based denoising solutions

Wiener filter [188] is one of the milestones in signal and image denoising and still nowadays is a crucial stage in state-of-the-art image denoising [191] and image processing [192] techniques. However, the introduction of Deep Learning dramatically outperformed most of the previous approaches making Convolutional Neural Networks (CNNs) a crucial step in many denoising algorithms [200–202]. Here, we tackle the denoising of mixed 2D signals, where, as detailed in Section 6.2.1, we have different

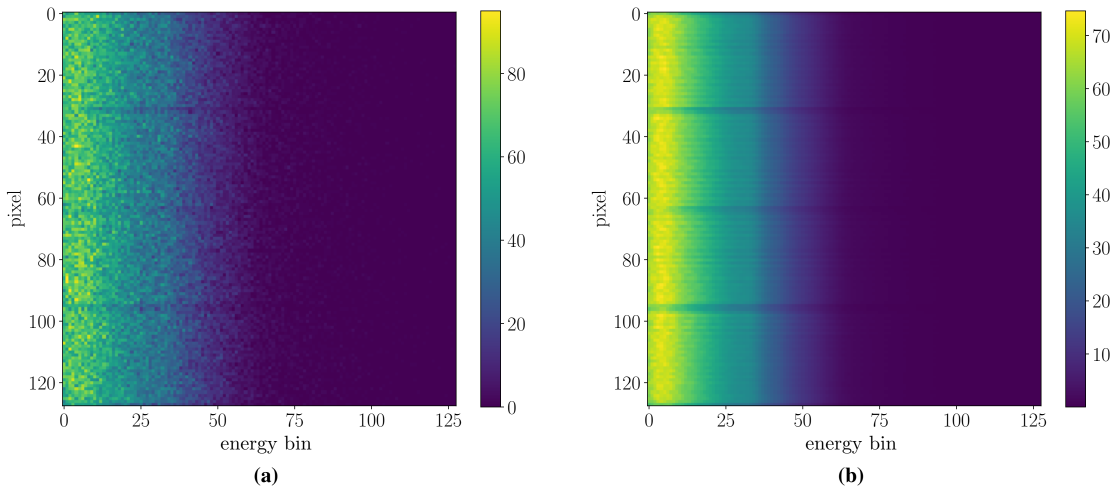


Figure 6.2: *(a) Single acquisition of background image. (b) Mean of 10000 acquisitions of background image.*

physical characteristics on the two axes (space and X-ray energy bin in our case). In particular, we compare the capabilities of the 2D Wiener filter to a CNN approach with the significant advantage that, for the latter, we do not need a detailed knowledge of noise and signal statistical behaviour. We follow an approach based on a CAE due to its proven effectiveness in many denoising applications [194, 201, 203–205] and the results provide an excellent agreement with the Wiener filter.

6.2.1 Problem formulation

Images acquired through digital acquisition systems are subject to different kinds of degradation. For the system under analysis, we can approximate noise as additive zero-mean noise due to thermal, shot and $1/f$ noise [206]. Formally we can define the acquired image $\mathbf{I}$ as:

$$\mathbf{I} = \mathbf{I}_0 + \mathbf{N}, \tag{6.4}$$

where $\mathbf{I}_0$ is the true (i.e., unaffected by noise) image signal and $\mathbf{N}$ is the additive noise term. With this model at hand, our goal is to estimate a denoised version of $\mathbf{I}$, namely $\hat{\mathbf{I}}$, as close as possible to $\mathbf{I}_0$, thus compensating for the detrimental effect of $\mathbf{N}$.

Under the assumption of additive zero-mean independent noise realizations affecting the acquisition, we can consider the sample mean $\bar{\mathbf{I}}$ of an adequate number of repeated acquisition (i.e., $10\,000$) of an image $\mathbf{I}$ as a good and not distorted estimate of the clean image $\mathbf{I}_0$. Therefore we consider $\bar{\mathbf{I}}$ as the ground truth for our algorithm's output (see Figure 6.2). However, note that the sample mean can only be applied in a controlled offline procedure, as it would not be feasible in a real-time system due to the necessity of multiple repeated acquisitions.

6.2.2 Preprocessing

Images directly obtained from the sensor are not suitable for denoising due to their dynamic range and the presence of banding artefacts. For this reason, we first apply two preprocessing operations before performing the denoising step.

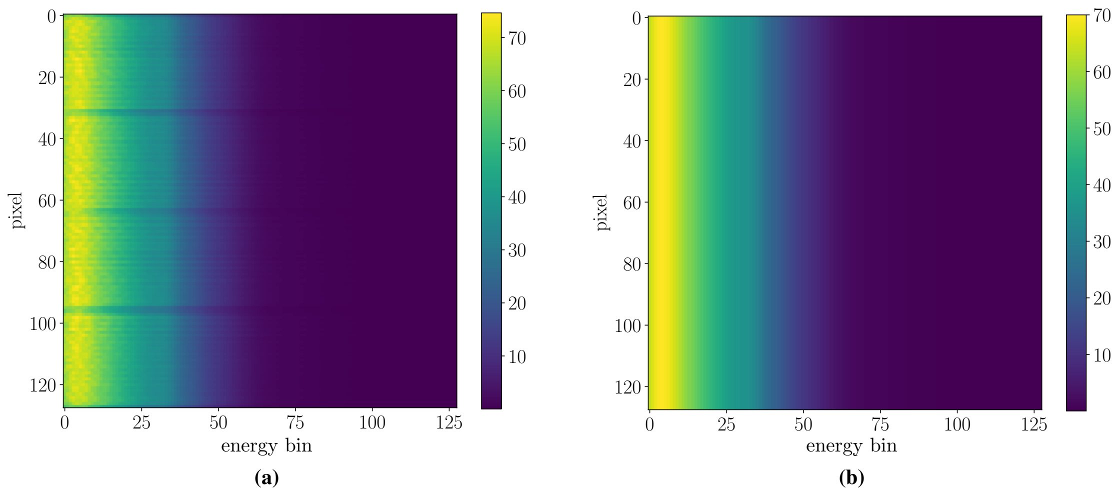


Figure 6.3: *Mean of background image. (a) Before row normalization. (b) After row normalization.*

Row Normalization

Figure 6.3a shows the average background image $\mathbf{M}$ obtained averaging $10\,000$ 'background' images (e.g., no objects were placed in front of the sensor). It is possible to notice that the average acquired signal shows three horizontal lines in which the values are significantly lower than the rest of the signal's rows. This effect is due to the form factor of the detector which is physically divided in four sub-sensors, each one contributing to 32 rows. To contrast this, we perform row normalization on each $H \times W$ input image $\mathbf{I}$, starting from the mean background signal $\mathbf{M}$. The 'background' signal is chosen as all the other acquired images have lower values (in the background image no objects intersect the X-ray path so the intensity of the detected photons is maximum).

The normalized version $\tilde{\mathbf{I}}$ of the input image $\mathbf{I}$ is obtained element-wise by:

$$[\tilde{\mathbf{I}}]_{ij} = \frac{[\mathbf{I}]_{ij}\,[\mathbf{m}]_j}{[\mathbf{M}]_{ij}}, \tag{6.5}$$

where:

$$[\mathbf{m}]_j = \frac{1}{H}\sum_{i=1}^{H}[\mathbf{M}]_{ij} \tag{6.6}$$

is the spatial average of $\mathbf{M}$ for frequency bin j. Figure 6.3b reports the normalized image $\tilde{\mathbf{I}}$. It is possible to notice that the periodic artefacts have been strongly attenuated.

Histogram Normalization

Another important preprocessing step consist in stretching the image dynamic in the range $[0, 1]$ in order to obtain an adequate input for the CNN. Since we are dealing with hyperspectral data, different images have different value distributions with their own maxima. We perform histogram normalization in order to obtain a more uniform distribution of data over the range $[0, 1]$ as shown in Figure 6.4. This is convenient for numerical stability in optimization algorithms, since it is more difficult reaching convergence with very skewed and imbalanced sample distributions.

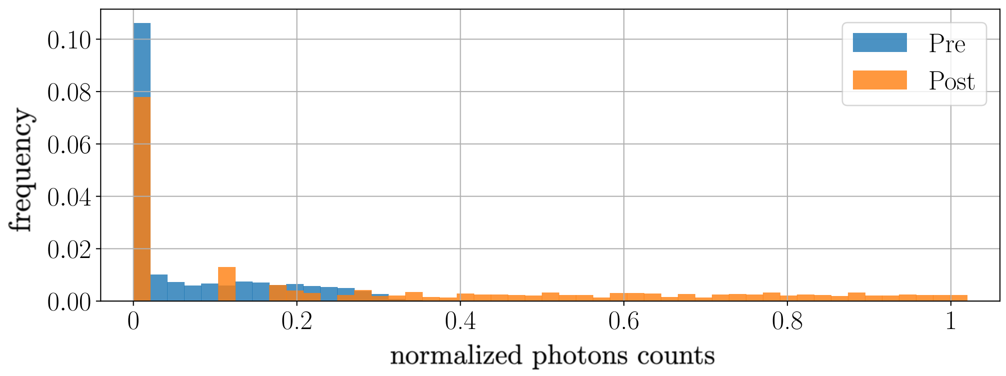


Figure 6.4: *Values distribution for an image pre and post histogram normalization.*

6.2.3 Denoising

After preprocessing, images are ready for the denoising step. We compare two different denoising strategies: a model-based solution based on Wiener filtering; a data-driven method based on the AE.

Wiener Filter

According to Equation (2.7), to apply a Wiener filter to an image, we need to estimate the power spectrum of the clean image $\mathbf{S}_I$ and the power spectrum of the noise $\mathbf{S}_N$. The power spectrum of the clean image $\mathbf{S}_I$ is computed as:

$$\mathbf{S}_I = |F_2(\bar{\mathbf{I}})|^2, \tag{6.7}$$

since the normalized mean image $\bar{\mathbf{I}}$ is considered representative of the clean image, where:

$$[\bar{\mathbf{I}}]_{ij} = \frac{1}{10000} \sum_{n=1}^{10000} [\tilde{\mathbf{I}}_n]_{ij}, \tag{6.8}$$

Following the same procedure, the power spectrum of the noise $\mathbf{S}_N$ is computed as:

$$\mathbf{S}_N = \frac{1}{10000} \sum_{n=1}^{10000} |F_2(\mathbf{I}_n - \bar{\mathbf{I}})|^2. \tag{6.9}$$

Convolutional AutoEncoder

The detailed architecture of the adopted CAE is depicted in Table 6.1. Given as input a noisy image $\mathbf{I}$, the CAE $C(\boldsymbol{\theta}, \mathbf{I})$ outputs directly the denoised version of the image $\hat{\mathbf{I}}$. The goal of the AE is to first reduce the dimensions of the image and then try to reconstruct it from this low-dimension representation, hopefully neglecting the noise affecting the original signal. The proposed architecture takes as input an image of shape $128 \times 128 \times 1$ and reduces it to $6 \times 6 \times 256$, applying a downsampling factor of almost 2.

SoftMax as the last layer's activation function ensures that the output lies in the $[0, 1]$ range.

Table 6.1: *Full list of CAE layers.*

Type	Filters	Activation	Output shape
Input	-	-	$(128 \times 128 \times 1)$
Convolution 2D	$32 \times (5 \times 5)$	ReLU	$(124 \times 124 \times 32)$
Max Pooling	(2×2)	-	$(62 \times 62 \times 32)$
Convolution 2D	$64 \times (3 \times 3)$	ReLU	$(60 \times 60 \times 64)$
Max Pooling	(2×2)	-	$(30 \times 30 \times 64)$
Convolution 2D	$128 \times (3 \times 3)$	ReLU	$(28 \times 28 \times 128)$
Max Pooling	(2×2)	-	$(14 \times 14 \times 128)$
Convolution 2D	$256 \times (3 \times 3)$	ReLU	$(12 \times 12 \times 256)$
Max Pooling	(2×2)	-	$(6 \times 6 \times 256)$
Upsampling	(2×2)	-	$(12 \times 12 \times 256)$
Deconvolution 2D	$128 \times (3 \times 3)$	ReLU	$(14 \times 14 \times 128)$
Upsampling	(2×2)	-	$(28 \times 28 \times 128)$
Deconvolution 2D	$64 \times (3 \times 3)$	ReLU	$(30 \times 30 \times 64)$
Upsampling	(2×2)	-	$(60 \times 60 \times 64)$
Deconvolution 2D	$32 \times (3 \times 3)$	ReLU	$(62 \times 62 \times 32)$
Upsampling	(2×2)	-	$(124 \times 124 \times 32)$
Deconvolution 2D	$1 \times (5 \times 5)$	SoftMax	$(128 \times 128 \times 1)$

6.2.4 Experimental Setup

In this section we report all details concerning the considered experimental setup. We first describe the considered dataset. We then report implementation details about the AE training procedure.

Dataset

The dataset is composed by acquisitions taken with MultiX ME100[1], which is a linear sensor of length 128 pixel. The operating point is 90keV and 0.2mA. For each acquisition, we place an object in front of the sensor, we irradiate it with a X-ray beam, and record the linear sensor output. For each object we acquire $10\,000$ static images (i.e., the item is steady in front of the sensor) changing the acquisition time $\tau = \{1\text{ms}, 1.5\text{ms}, 2\text{ms}\}$ (i.e., the active recording time of the sensor, similar to a camera shutter time).

We divide the energy spectrum of the acquisition in 128 bins. The final shape of each measure is therefore 128×128, where the first dimension is the number of pixels in a single column, so acquired in the vertical direction, and the second is the number of energy bins used to represent the spectrum. Different plastic polymers of different thickness are considered as objects, as listed in Table 6.2. These polymers have a chemical composition which represents the most common contaminants that can be found during food inspections. For each material width δ and for each acquisition time τ three measures have been done, totalling 138 measures. The whole corpus of images is therefore composed of $1\,380\,000$ hyperspectral images of shape 128×128.

As both Wiener filtering and the AE rely on the offline estimation of a set of parameters (i.e., power spectrums and network weights), we applied a commonly adopted training-validation-testing split policy to avoid biasing the achieved results. We paid at-

[1]https://www.qualityassurancemag.com/article/multix-me100-x-ray/

Table 6.2: *Considered materials for the acquisitions.*

Material	**Label**	**Width δ [mm]**
Background	BKG	-
Polyethylene	PE	$2, 8, 16$
Polyamide	PA66	$2, 8, 16$
Polyoxymethylene	POM	$2, 8, 16$
Polytetrafluoroethylene	PTFE	$2, 8, 16$
Polyvinyl chloride	PVC	$2, 8, 16$

tention to keep in the same set measurements of the same material (including width δ) with the same acquisition time τ. Hence, we randomly use 70% of the data for training (of which 21.4% for validation) and 30% for testing.

Training and testing pipeline

We develop our AE model using Keras with TensorFlow as backend [207]. The network is trained to minimize the loss function $\mathscr{L}$ computed on batches of $K = 50$ noisy images and their denoised version $\hat{\mathbf{I}}$:

$$\mathscr{L} = \frac{1}{K}\sum_{k=1}^{K} ||\mathbf{I}_k - \hat{\mathbf{I}}_k||_F^2 = \frac{1}{K}\sum_{k=1}^{K} ||\mathbf{I}_k - A(\boldsymbol{\theta}, \mathbf{I}_k)||_F^2, \tag{6.10}$$

where $||\cdot||_F$ denotes the Frobenius norm. We use Stochastic Gradient Descent (SGD) with Nesterov momentum [53] as optimizer, setting initial learning rate to 0.01, learning rate decay to 10^{-6} each epoch and momentum to 0.9, until reaching convergence on a validation plateau. At testing time, we freeze the network weights and we compute the output from each image in the testing dataset.

6.2.5 Results

Table 6.3 shows the results for the two considered denoising algorithms in term of Peak Signal-to-Noise Ratio (PSNR) , which is defined as:

$$\mathrm{PSNR} = 20 \cdot \log_{10}\left(\frac{HW}{||\mathbf{I} - \hat{\mathbf{I}}||_F^2}\right), \tag{6.11}$$

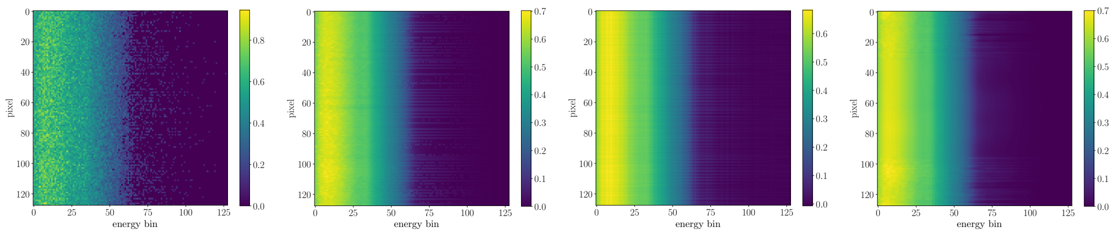
where $\mathbf{I}$ and $\hat{\mathbf{I}}$ are the compared images of size $H \times W$ whose range is $[0, 1]$, and the numerator term HW is needed to compensate for the signals' size.

Even though both algorithms yield good results, it is worth noting that the implementation of Wiener filter is heavily tailored to the considered acquisition system data model. Nevertheless, PSNR between the two methods is comparable, showing that it is possible to work in a CNN fashion for hyperspectral X-ray denoising.

In particular, the advantage of using the AE is that we do not need to have any kind of prior knowledge on the input data model, and we only need some training images. This leads to better generalization capability with respect to Wiener filter method, which requires a carefully characterization of the Noise-to-Signal ratio $\mathrm{S}_N / \mathrm{S}_I$.

Table 6.3: *PSNR for the considered methods. Best results in bold.*

Material (δ, τ)	Wiener filter PSNR [dB]	AE PSNR [dB]
BKG 2ms	**41.192**	38.729
PA66 16mm 1.5ms	**40.827**	**40.827**
PA66 2mm 1.5ms	**39.720**	39.406
PE 2mm 1.5ms	**39.259**	38.852
PE 8mm 1.5ms	**39.830**	39.721
POM 16mm 2ms	**41.329**	40.880
POM 2mm 2ms	**40.479**	39.371
POM 8mm 1.5ms	**40.362**	40.163
PTFE 16mm 1ms	**41.696**	40.300
PTFE 2mm 1.5ms	**39.900**	39.626
PTFE 8mm 2ms	**40.605**	40.015
PVC 16mm 1ms	**40.897**	37.289
PVC 16mm 2ms	**42.649**	40.605
PVC 2mm 2ms	**40.589**	39.521



(a) *Input* (b) *Ground truth* (c) *Wiener filter* (d) *CAE*

Figure 6.5: *Output images from the considered denoising methods.*

Moreover, the good denoising results achieved by the CAE could be further exploited in case supervised problems (e.g., contaminant detection, segmentation, classification) should be faced after denoising. As a matter of fact, a transfer learning approach [208] could be adopted in order to develop and train different CNNs that share many layers with the one proposed here, dramatically reducing the number of parameters to be trained. A visual example of the goodness of output images with respect to the ground truth is shown in Figure 6.5 and a per-pixel plot is depicted in Figure 6.6.

An interesting aspect to analyze is the output of the hidden layers of the trained network. In fact, output from hidden encoding layers displays the automatically extracted features from the input image, giving us a clue on what the network is highlighting during the encoding procedure. Figure 6.7 displays some of the output from the innermost hidden layer, generated by using as input a sample from the measure “PVC 8mm at 2ms”. We can clearly notice different neurons activating at different energy bins, each one capturing different shapes and intensities that will be used in the decoder to reconstruct the final output. This enables us to consider the CAE as a methodology learning a set of meaningful basis for data projection.

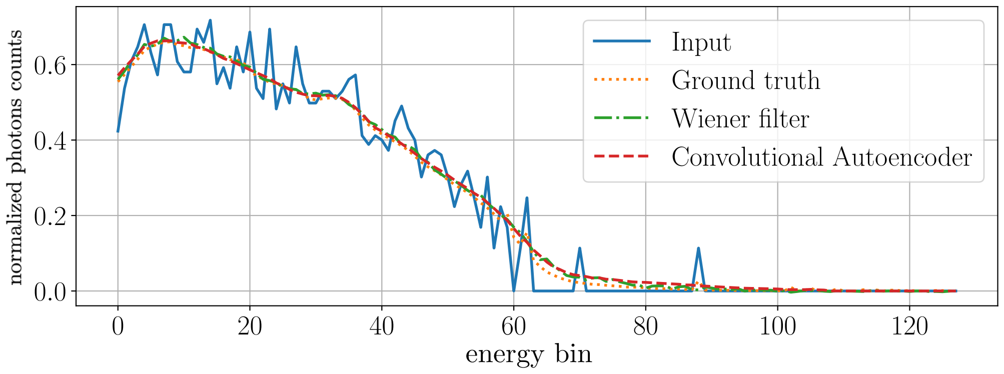


Figure 6.6: *Comparison of different methods on row 50 of a specific image. The trend on other images is the same.*

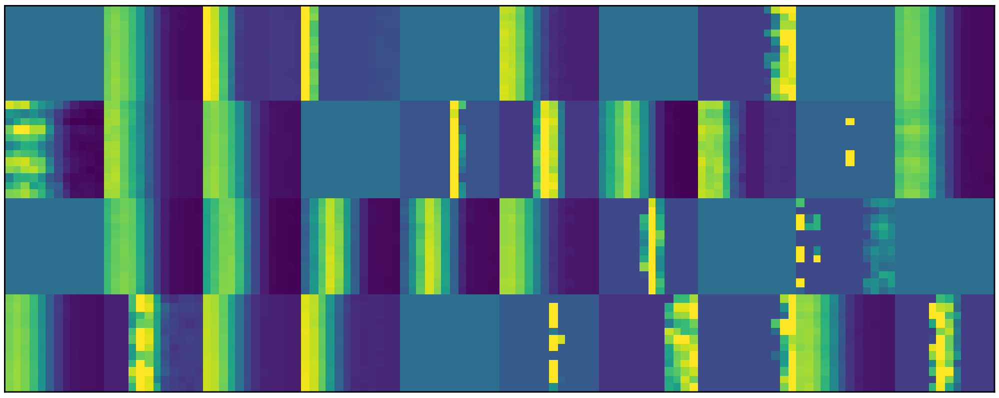


Figure 6.7: *Hidden layers representation.*

6.3 Comparison between different AutoEncoder architectures

In this section we propose a comparison between AEs and VAE for the tasks of hyperspectral acquisition denoising. We train the networks to reconstruct the noise-free version of a given noisy acquisition in an unsupervised fashion, without using the average signal as a groundtruth. We show how the networks can denoise the input even in this challenging scenario.

6.3.1 Problem Statement

The acquisition system we consider is depicted in Figure 5.1. The whole system is mounted around a conveyor belt directly on the production's site, and it performs real-time acquisitions of food containers. It is composed by an X-ray generator emitting the X radiation registered by the hyperspectral detector after travelling through the food container. The detector counts the photons which have not been absorbed by the scanned object at different spectral frequencies. In this framework, frequencies are commonly named X-ray energies due to the Planck-Einstein Relation [184]. The detector is a linear sensor array composed by P pixels which samples the space in different positions. The acquired spectrum is divided into B photon energy intervals, named "energy bins" from now on. The complete output of the sensors is therefore a

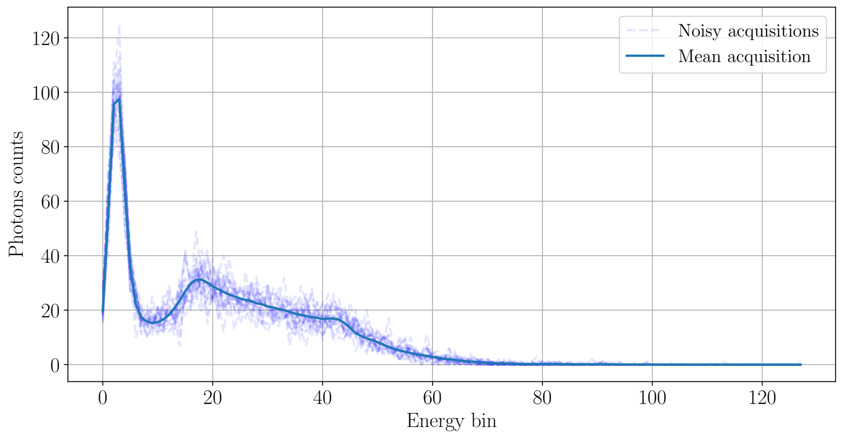


Figure 6.8: *20 noisy acquisitions with their mean.*

$P \times B$ matrix of photon counts for each time instant. We call $\boldsymbol{y}$ the B-dimensional vector containing the photons count for each energy bin of a single pixel in a certain time instant. Notice that, differently from the previously proposed method, we are now considering 1D signals (i.e., the output of the sensor for a single pixel). We made this choice for the sake of simplicity, without undermining the generality of the method, which can be simply extended by considering more than one pixel at a time in a 2D signal.

Due to the stochastic nature of photon's emission, the acquisition $\boldsymbol{y}$ is highly affected by noise. We could approximate the clean acquisition, named $\boldsymbol{\lambda}$ from now on, by averaging together an adequate number of noisy acquisitions of the same object, as depicted in Figure 6.8. This is not feasible in the system's scenario, with the conveyor belt moving in the range of $1\,\mathrm{m/s}$ and the object intersecting the X-ray beam for just a split second (a typical acquisition time is $0{,}2\,\mathrm{ms}$). Due to this time constraint, we cannot always measure $\boldsymbol{\lambda}$ in our analysis and we are forced to develop a time-efficient solution which does not rely on it. Our ultimate goal is to reduce the acquisition time with the minimum possible impact on a subsequent contaminant detector. In our experiments, we consider a single pixel and 128 energy bins, therefore the output of the acquisition system will have a shape of 1×128. It is worth noting that selecting a single pixel reduces the computational cost without affecting the generalization of the method, as every pixel has to be considered separately due to constructive characteristics. Moreover, a single pixel is the smallest possible detectable dimension for a contaminant, giving us insights on the most difficult scenario we can encounter.

We consider the problem of estimating the clean version of a noisy acquisition $\boldsymbol{y}$, namely $\boldsymbol{\lambda}$. More formally:

$$\hat{\boldsymbol{\lambda}} = \mathrm{N}_d\left(\boldsymbol{y}\right), \tag{6.12}$$

where $\hat{\boldsymbol{\lambda}}$ is the estimation of $\boldsymbol{\lambda}$ produced by the operator N_d. Our goal is to design the operator N_d. Notice that we do not make use of the actual $\boldsymbol{\lambda}$ in the equation, as we want to develop an unsupervised method.

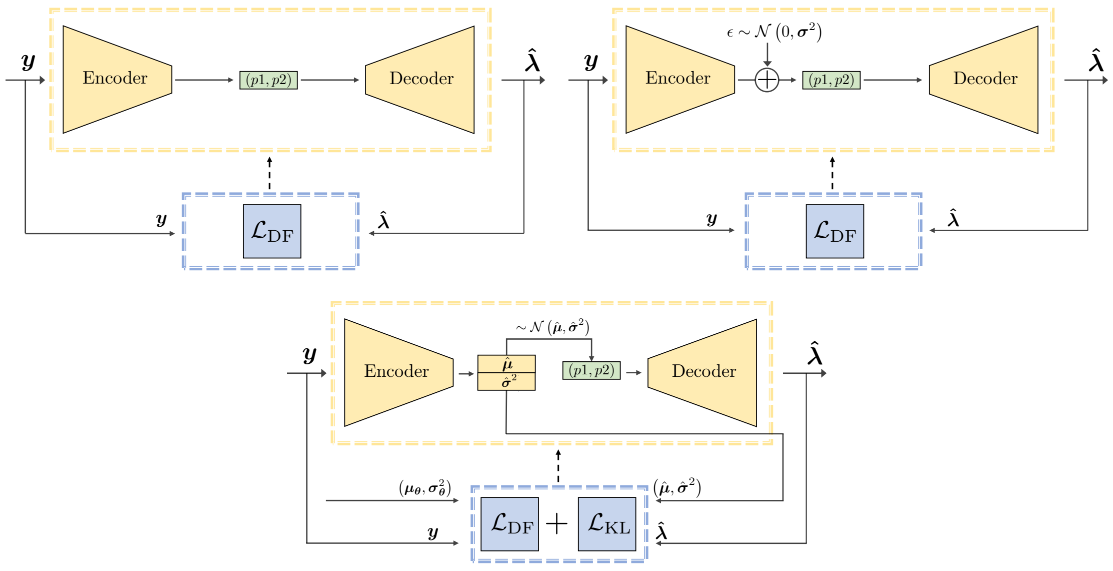

Encoder
$(p1, p2)$
Decoder
$\boldsymbol{y}$
$\hat{\boldsymbol{\lambda}}$
$\mathcal{L}_{\mathrm{DF}}$
$\epsilon \sim \mathcal{N}(0, \sigma^2)$
$\sim \mathcal{N}(\hat{\mu}, \hat{\sigma}^2)$
$\hat{\mu}$
$\hat{\sigma}^2$
$(\mu_\theta, \sigma_\theta^2)$
$(\hat{\mu}, \hat{\sigma}^2)$
$\mathcal{L}_{\mathrm{DF}} + \mathcal{L}_{\mathrm{KL}}$

Figure 6.9: *Schematic representation of the three considered architectures. Top left: AE, top right: AuAE, bottom: VAE.*

6.3.2 Proposed Method

We propose three different network architectures for implementing the system, depicted in Figure 6.9. In this section we describe the network architectures and the loss functions we use for training.

Network architectures

As stated before, we implement the system with three particular network architectures (i) AE, (ii) AuAE, (iii) VAE.

We choose the AE given the good denoising results such a network provides in [28, 29], despite its simplicity. The AE is composed by an Encoder and a Decoder. During the encoding, the input signal dimensionality of 128 is halved 6 times by a sequence of Fully Connected (FC) layers until reaching the latent space dimensionality of 2. The choice of setting the latent space's dimension to 2 is motivated by the Lambert-Beer law (Section 5.3). Exponential Linear Unit (ELU) activation function is employed after each FC layer. The decoding phase is symmetrical to the encoding, with the latent space being doubled 6 times up to the original dimensionality of 128. In addition to the ELU, we perform a light Dropout right after each FC layer, to prevent overfitting [209]. The last activation is a Leaky ReLU [210] as we want to penalise the output being negative.

The AuAE is an AE whose latent space is injected with Gaussian noise. It follows the very same structure of the AE. Once the latent space has been encoded, it is perturbed by a Gaussian noise centered into the encoded value and with a learnable standard deviation. This means that the network itself learns to set the most appropriate noise to minimize the loss function. In other words, the network is asked to jointly learn the best encoder/decoder weights to filter out the input noise, and the augmentation noise to cast on the latent space to provide more robustness to the whole procedure.

The VAE is chosen for its capability of constraining the distribution of the latent space. Its structure is the same of the AE, except for the last FC layer of the encoder which is replaced by two separate layers needed to estimate the parameters of the latent space distribution. The actual latent space is retrieved by sampling from this distribution, following the reparameterization trick described in [196].

In the next sections we will refer to the latent space of a network as (p_1, p_2), being the two parameters p_1 and p_2 learned by the network.

Loss functions

The adopted loss function depends on the architecture at hand. In this section we formally define the loss function for the i-th data sample. The unsupervised AE and AuAE are trained by minimising the MSE between the noisy input acquisition $\boldsymbol{y}$ and the output $\hat{\boldsymbol{\lambda}}$:

$$\mathscr{L}_{AE} = \mathscr{L}_{DF} = \frac{1}{B} \sum_{b=1}^{B} \left(y_b - \hat{\lambda}_b \right)^2, \tag{6.13}$$

where B is the number of considered energy bins. Notice that the network is forced to output a clean version of the signal without seeing the actual reference mean spectrum. This particular employment of MSE is motivated by recent works in image denoising [211, 212], which already proved successfully in our previous work [29].

Regarding the VAE loss function, we need to take into account the prior distribution we impose on the latent space. In our case we adopt a multivariate Gaussian prior $p_{\boldsymbol{\theta}} = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with a diagonal covariance matrix $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1^2, \sigma_2^2)$. Following the usual formulation proposed by [196] we define $\mathscr{L}_{VAE}$ as:

$$\mathscr{L}_{VAE} = \mathscr{L}_{DF} + \mathscr{L}_{KL}, \tag{6.14}$$

where $\mathscr{L}_{DF}$ is a data fidelity term represented by the MSE in the same way of Equation (6.13) and $\mathscr{L}_{KL}$ is the Kullback-Leibler divergence between the distribution of latent space parameters, $q_{\boldsymbol{\phi}}$, and the prior: $D_{KL}(q_{\boldsymbol{\phi}} || p_{\boldsymbol{\theta}})$. We can compute $\mathscr{L}_{KL}$ analytically for the multivariate Gaussian case:

$$\mathscr{L}_{KL} = \frac{1}{2} \sum_{j=1}^{J} \left[\log\left(\frac{\sigma_{\theta_j}^2}{\hat{\sigma}_{\phi_j}^2} \right) + \frac{\left(\hat{\mu}_{\phi_j} - \mu_{\theta_j}\right)^2}{\sigma_{\theta_j}^2} + \frac{\hat{\sigma}_{\phi_j}^2}{\sigma_{\theta_j}^2} \right] - 1, \tag{6.15}$$

where J is the dimensionality of the latent space. For the sake of brevity, we just described the derivation of the framework we need to use the VAE for our purpose. We point to the original VAE paper [196] and its appendixes for a detailed explanation of the mathematical background.

6.3.3 Experiments

In this section we discuss the experimental setup we adopted to validate our method. We describe the dataset and the results we collected after performing the experiments. For what concerns the implementation, Pytorch framework is used for building and training the whole pipeline. All the experiments are run on a workstation equipped with an NVIDIA Titan V Graphics Processing Unit (GPU), an Intel Xeon E5-2687W and 256 GiB RAM.

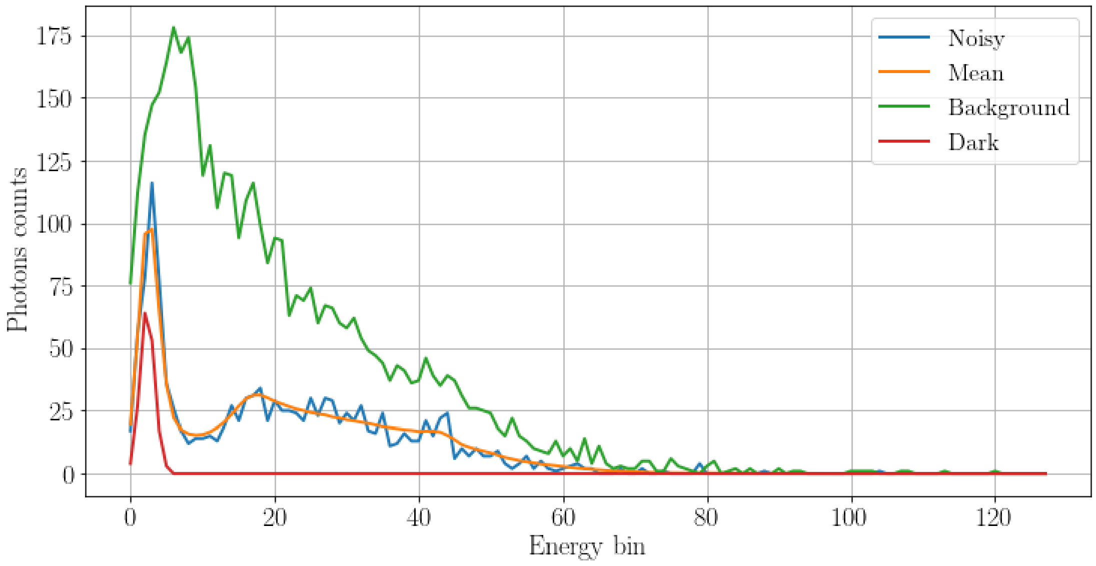


Figure 6.10: *Examples of acquisitions. Blue: noisy acquisition $\boldsymbol{y}$; orange: mean acquisition $\boldsymbol{\lambda}$; green: 'background' acquisition; red: 'dark' acquisition.*

Dataset

The dataset is composed by acquisitions from the system, as mentioned above. The X-ray tube operates at 100kV and 0.2mA. We place each object in front of the sensor and irradiate it with an X-ray beam, recording its output. We divide the energy spectrum of the acquisition into $B = 128$ energy bins. Each object is statically acquired 2555 times with an acquisition time of 2ms. Due to statistical noise in photons counts, each acquisition of the same object differs from the other. By averaging together all the acquisitions of the same object, we obtain a good approximation of the "real" spectrum $\boldsymbol{\lambda}$ of that object.

The dataset is composed by 8 plastic polymers: 'PA', 'PC', 'PE', 'PET', 'PMMA', 'PTFE', 'PVC', 'PVDF', 2 metals: 'Al', 'Cu' and acquisitions of 'background' (i.e., the source's X-ray directly acquired by the sensor without objects in the middle) and 'dark' (i.e., the signal acquired by the sensor with the X-ray source switched off). Figure 6.10 shows an example of a noisy spectrum $\boldsymbol{y}$, alongside its clean version $\boldsymbol{\lambda}$. In the same plot we also show for reference an example of 'background' spectrum and one example of 'dark' spectrum.

Every material (associated with a specific parameter Z_{eff}) is represented by tiles of different thicknesses (associated with the parameter $\rho\Delta x$). The thicknesses range from 2mm to 96mm, for a total of 93 different combinations of Z_{eff} and $\rho\Delta x$ (i.e., the different classes). Figure 6.11 shows a simplified representation of 10 random classes. We divide the 128 energy bins into three macrobins, from energy bin 0 to 20, from 20 to 40 and from 40 to 128. The uneven spacing accounts for the generally decreasing photon counts at the higher energies. We then sum the photons counts inside each macrobin to obtain a 3D representation. We can notice that some classes have significant overlaps.

We experimentally verified that the sensor's statistical noise follows a Poisson distribution, independent over each energy bin, with rate equal to the mean photons count for that specific bin. We use this information to augment the quantity of data by gen-

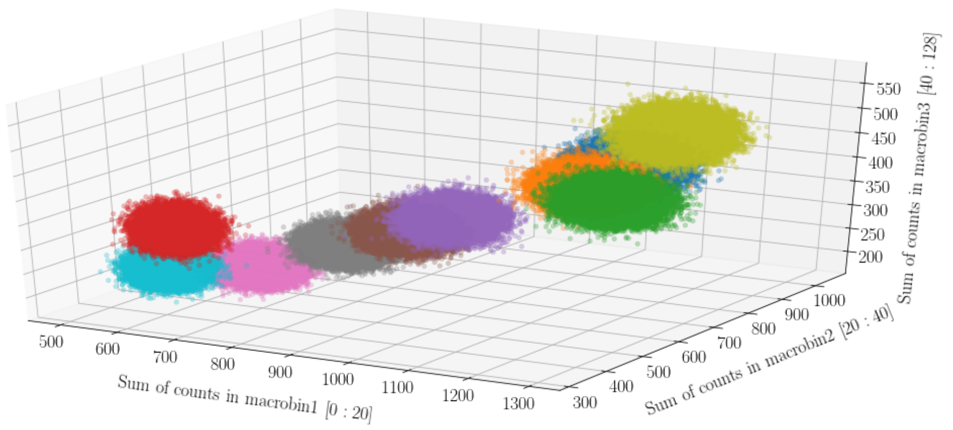


Figure 6.11: *Simplified representation of* 10 *random input materials. The* 128 *energy bins are divided into three macrobins. In each macrobin, the total number of photons counts are summed, to obtain a 3D projection of the input spectra. Each color represents a different class.*

erating synthetic acquisitions from the experimental mean spectra $\boldsymbol{\lambda}$. It is worth noting that this synthesis method does not introduce new materials to the list of the acquired ones, it just expands the number of available samples for each material. For each class, we generate $110\,000$ noisy acquisitions, except for background and dark, for which we generate 2.5 million noisy acquisitions to balance their frequency with respect to the rest of the classes. The dataset is therefore composed by roughly 15 million noisy acquisitions. We split this amount into a train (70%) and a test set (30%), further reserving the 20% of the train set for validation. As a last additional step, we remove the 'background' and 'dark' acquisitions from the test set, as we would like to evaluate the performances of the methods in the more realistic scenario in which we always have some material passing through the system.

Metrics

Evaluating the denoising capability of the system is a matter of defining a "similarity" metric between the true clean spectrum $\boldsymbol{\lambda}$ and the output of the network $\hat{\boldsymbol{\lambda}}$. As stated above, we know that the photons' counts noise on every energy bin follows an independent Poisson distribution. We can therefore use a Weighted Log-Likelihood (WLL) as a metric to evaluate the fidelity of the network output with respect to the input:

$$\mathcal{J} = -\frac{1}{B}\sum_{b=1}^{B}(\lambda_b(\ln(\hat{\lambda}_b) - \ln(\lambda_b)) - (\hat{\lambda}_b - \lambda_b)). \tag{6.16}$$

The quantity $\mathcal{J}$ gives us a scalar measure of the compatibility between the predicted spectrum and the real one, and is the same metric we already used in [29]. For the sake of clarity, we further define:

$$\mathrm{A}_y = \mathrm{e}^{-\mathcal{J}}, \tag{6.17}$$

which bounds the compatibility between 0 (poor result) and 1 (perfect match), and can be interpreted as the accuracy of the prediction.

Table 6.4: *Results in terms of different metrics for the synthetic dataset with no background and no dark (best in bold).*

Model	A_y	$\mathrm{A}_{\mathrm{SCNR}}$	MSE
AuAE	0.9924	0.9698	1.4253
VAE_μ	**0.9926**	**0.9710**	**0.7740**
$\mathrm{VAE}_{\mathrm{avg}}$	0.9925	0.9704	0.7876
AE	0.9586	0.7689	5.3618
PCA_2	0.9502	0.8135	6.5185

Table 6.5: *Results in terms of different metrics for the real dataset (best in bold).*

Model	A_y	$\mathrm{A}_{\mathrm{SCNR}}$	MSE
AuAE	**0.9927**	0.9709	1.3829
VAE_μ	0.9926	**0.9715**	**0.7227**
$\mathrm{VAE}_{\mathrm{avg}}$	0.9801	0.9694	0.7371
AE	0.9588	0.7681	5.3618
PCA_2	0.9318	0.7829	6.4500

We define another metric for assessing the denoising capability of a network, the Spectral Contrast to Noise Ratio (SCNR):

$$\mathrm{SCNR} = \frac{1}{B}\sum_{b=1}^{B}(\ln(\hat{\lambda}_b) - \ln(\lambda_b))(\hat{\lambda}_b - \lambda_b), \tag{6.18}$$

which measures how much the prediction can be distinguished from the groundtruth. A SCNR equal to 0 means that the two signals are indistinguishable, while a higher value indicates that the prediction differs more from the groundtruth. Similarly to Equation (6.17), we can further define:

$$\mathrm{A}_{\mathrm{SCNR}} = \mathrm{e}^{-\mathrm{SCNR}}, \tag{6.19}$$

which is bounded between 0 (poor result) and 1 (perfect match).

As a third metric, we also evaluate the pure MSE between the networks output and the groundtruth:

$$\mathrm{MSE} = \frac{1}{B}\sum_{b=1}^{B}\left(\lambda_b - \hat{\lambda}_b\right)^2, \tag{6.20}$$

which is the same metric adopted as a loss function during training, with the exception that in Equation (6.13) we use the network input $\boldsymbol{y}$ as reference instead of the groundtruth $\boldsymbol{\lambda}$.

6.3.4 Results

We report the results in terms of A_y, $\mathrm{A}_{\mathrm{SCNR}}$ and MSE in Table 6.4. To better take into account the stochastic nature of the VAE, we decided to evaluate its performance in two different testing scenarios:

1. $\mathrm{VAE}_{\mathrm{avg}}$ considers as network output $\hat{\boldsymbol{\lambda}}$ the average of 100 predictions for each input $\boldsymbol{y}$. This mitigates the uncertainty of the encoding, at the cost of increasing

the prediction time. We experimentally choose 100 as the number of averaging acquisitions as it offers a good trade-off between accuracy and time.

2. VAE_{μ} considers the estimated mean as the latent space sample to be decoded. In other words, we do not consider the uncertainty of the encoding, and the VAE becomes a deterministic architecture.

Notice that this two scenarios are considered only when testing the network: the training procedure is exactly the same for the two of them and involves random sampling from the encoded distribution. We also report in the table the accuracy obtained with a PCA_2, as a baseline. This involves finding the first 2 principal components from the noisy train acquisitions, and projecting the test set onto them to obtain the reconstruction.

We also repeat every experiment of Table 6.4 on the dataset of real acquisitions we used for the generation of the synthetic one. We report those results in Table 6.5. Apart the Principal Component Analysis (PCA), which benefits from the reduced number of background and dark acquisitions, the metrics do not change significantly in this scenario, meaning that the proposed solution can be applied to real data without further adaptation. A visual reference of the denoising capabilities of each method is reported in Figure 6.12.

Latent Space

Figure 6.13 shows the distributions for 10 randomly selected materials over the (p_1, p_2) latent space. We notice how the AE tends to map every acquisition to a straight line which suggests a strong correlation between p_1 and p_2. AuAE and VAE_{μ} are less prone to squashing the latent space, and we can recognize the gaussian distributions in the latter. PCA_2 does not offer a latent space separation among the different materials.

Ablation tests

As a final set of experiments, we perform an ablation test on two hyperparameters of the AE: (i) the dimension of the latent space, (ii) the adopted activation function. We run this tests only on the AE as all the three considered networks share the very same encoder and decoder structure, so we expect very similar results also for the other two considered networks.

Regarding the latent space, we initially choose a dimension of 2 driven by the physical parameterization explained in Section 5.3. To understand how the final accuracy is affected by this choice, we modify the AE structure to have several increasing dimensions of latent space: $1, 2, 3, 4, 8$. To do so in a fair way and keep the total number of network parameters approximately constant, we also have to remove one intermediate layer at a time in both the encoder and the decoder side, up to three. As a result, we have 14 different combinations of number of intermediate layers and dimensionality, as depicted in Table 6.6. Table 6.6. As a side effect, also the number of parameters in the final layer of the encoder (and the first of the decoder, symmetrically) changes, as reported in the first column.

We express this metric in dB with respect to the best result obtained with latent space equal to 2, as comparison with the architecture shown in Table 6.4. We can notice how all the configurations perform worse than the one with latent space equal to 2. It is

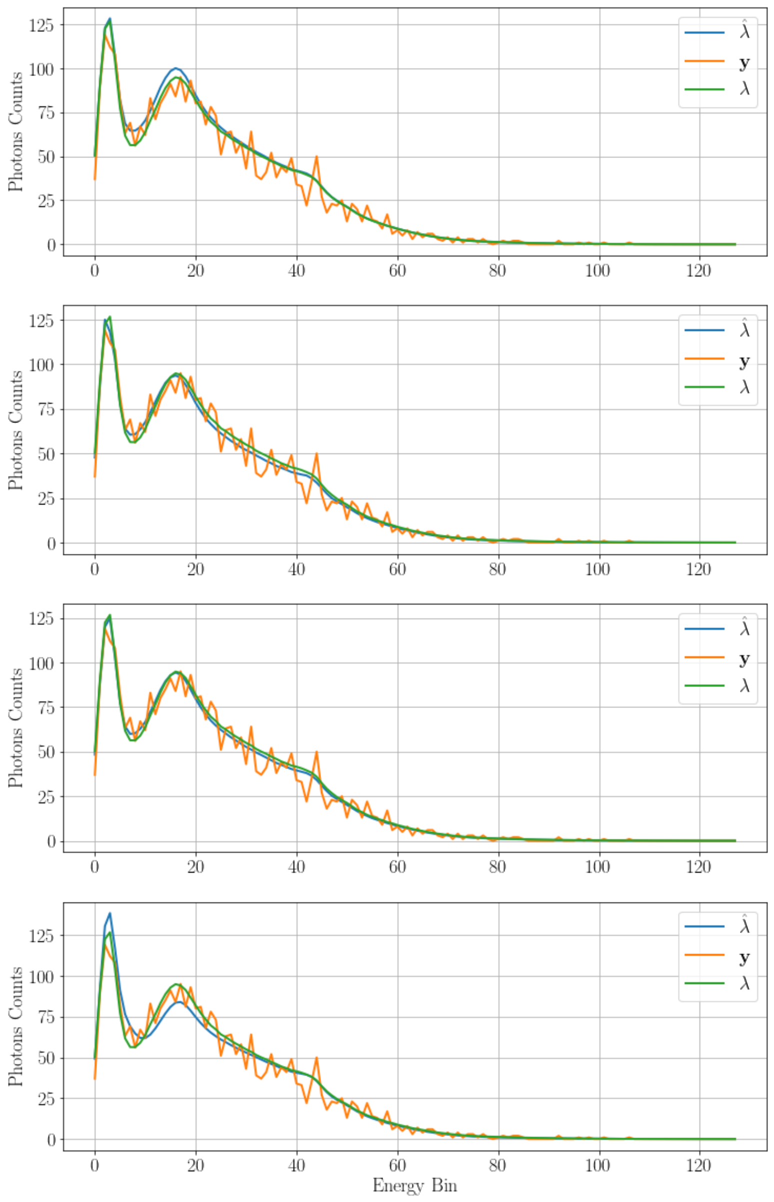


Figure 6.12: *Example of denoised output (blue) provided by different methods for the same input (orange) compared with the groundtruth (green). From top to bottom: AE, AuAE,* VAE_{μ}, PCA_2.

reasonable to think that the larger the latent space, the better the network's capability to find the best possible inner representation for a given input. What we show here somehow go against that intuition and justifies our choice of defining a latent space with just 2 parameters to force the network to learn a physically plausible reduction. We do not increase the latent space beyond 8 to avoid losing the meaning of signal compression and risking overfitting.

Regarding the non-linearity used as activation function in the intermediate layers of the AE, in addition to the original ELU we try other two options: ReLu and Leaky ReLU. We decided to keep the same Leaky ReLU function as the final activation, for penalising negative output without run into stability problems. Table 6.7 reports the

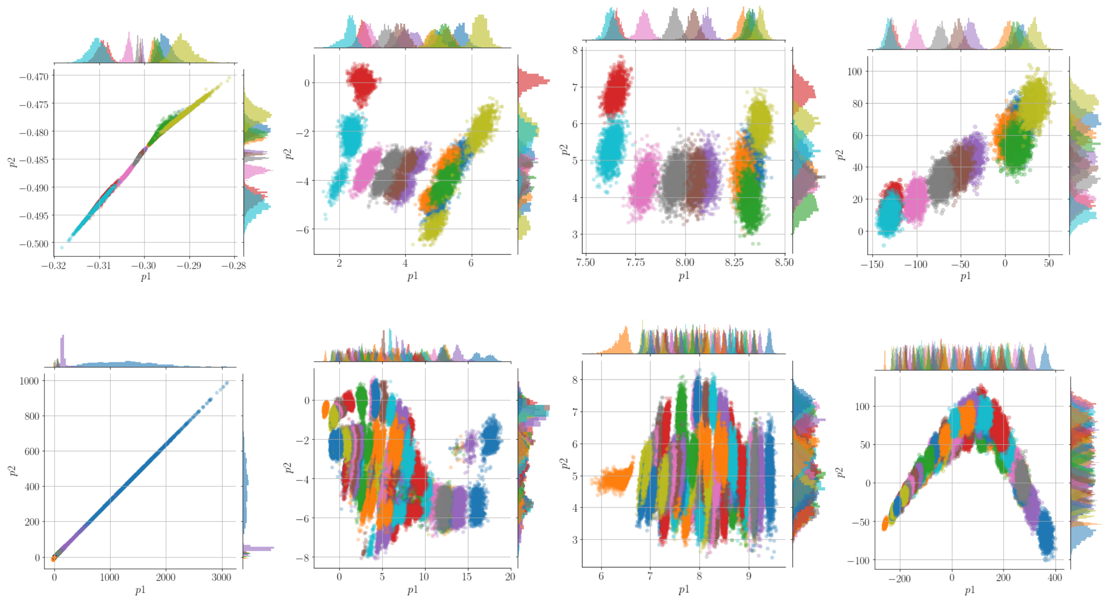


Figure 6.13: *Latent space distribution of parameters* p_1, p_2 *for* 10 *random materials, first row, and all the testing dataset, second row. Different classes are displayed with different colors. From left to right: AE, AuAE,* VAE_μ, PCA_2. *The histograms show the marginal distributions of* p_1 *and* p_2.

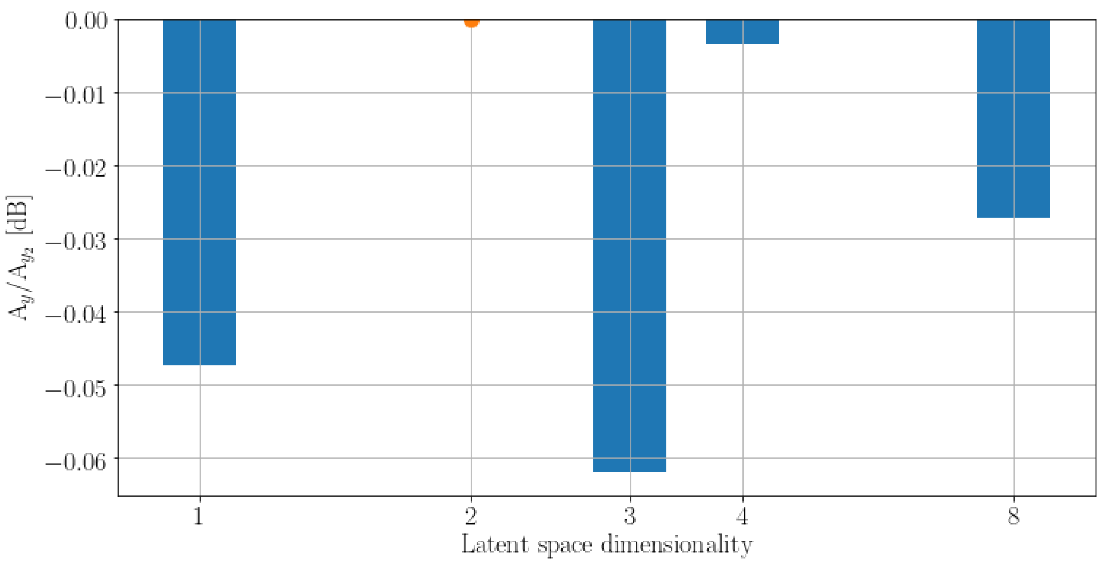


Figure 6.14: *Results in terms of* A_y *varying the latent space dimensionality. The y-axis is expressed in dB with respect to the* A_y *reached by the network with latent space equal to* 2.

results in term of A_y for the three options. The combination of ELU and Leaky ReLU we chose in first place reveals the best option among the three.

Table 6.6: *Network configurations for the ablation test on latent space dimensionality.*

# final neurons	latent dimension	# params
16	1	21 889
8	1	22 145
4	1	22 209
16	2	21 922
8	2	22 162
4	2	22 218
16	3	21 955
8	3	22 179
4	3	22 227
8	4	22 196
16	4	21 988
4	4	22 236
16	8	22 120
8	8	22 264
16	16	22 384

Table 6.7: *Results in terms of A_y for different activation functions in the intermediate layers of AE's encoder and decoder. Best in bold.*

Activation	A_y
ELU	**0.9602**
Leaky ReLU	0.9211
ReLU	0.8982

6.4 Conclusions

In this chapter we investigated different denoising strategies for X-ray hyperspectral images. We considered both model based (i.e., Wiener filter, PCA) and data driven (i.e., AE, CAE, VAE, AuAE) techniques.

We did a first comparison between the Wiener filter and the CAE. The considered pipeline is composed of a preprocessing step necessary for normalizing input images, followed by candidate denoising solutions. Results show that the solution based on AE provides close results to the one based on Wiener filter. This is interesting as it is a strong indicator that the proposed AE is capable of learning a meaningful reduced dimensionality representation of the input data.

Starting from this result, we pushed the limit of the AEs family by reducing the dimension of the latent space. We started from the assumption that a possible parameterisation of the spectra uses just two parameters, thanks to the Lambert-Beer law. We validate our method on an extensive dataset of plastic polymers and metals, synthesised starting from real acquisitions. Results show how all the networks can filter and reconstruct the mean spectrum starting from the noisy acquisitions in an unsupervised fashion without having access to the groundtruth. All of them reach excellent results despite the simplicity of the structure and the reduced dimension of a latent space of just two parameters.

Nonetheless, the AuAE proved the more flexible network, as it shows the best results in terms of the designed denoising metrics, and it has a deterministic nature once trained. Given these results, we can consider the AuAE as the best option for real time denoising of Hyperspectral X-ray.

We considered single polymer acquisitions as a simplification of the general case of X-rays crossing a section of composite material (e.g., the food and the contaminant). The Lambert-Beer law ensures that we can make the assumption of a single homogeneous material without loosing generalization. In practice, we cannot directly apply the Lambert-Beer law to the X-ray acquisition for classification purposes. In fact, the law holds for mean spectra but we are measuring single acquisitions that suffer from statistical noise. Moreover, a direct use of the law would require a linear and invertible sensor function, which is not the case due to non-linear effects (i.e., pile-up) and structural imperfections of the sensor. This motivates the need of denoising in the acquisition pipeline.

A possible limitation of the proposed method lies in the statistical modelling of the noise affecting the acquisition. Should the Poisson statistic in photons counts not be verified anymore, this method could suffer from the discrepancy between training and testing data, possibly requiring a retraining on that specific data. In this regards, the proposed method is fitted on the measurement setup.

A future work to target this problem will investigate the impact of training the network with background acquisitions as an additional input. This would in principle help the network to 'normalize' the input, learning setup-independent features and improving the generalization capability under discordant training/testing scenarios. Another future work will be devoted to finding a suitable procedure to map the latent space of the network to the actual physical parameters associated with the acquired material by the Lambert-Beer law. Besides, we can further expand the methodologies presented in this chapter and develop other applications built on top of the denoising algorithm (e.g., food contaminant detection, anomaly detection, material recognition, etc.). Indeed, it could be possible to exploit the deep representation of the input data provided by the AE as candidate denoised feature vector to be used for classification purpose.

CHAPTER 7

Hyperspectral X-ray polymer classification

An intriguing problem of *physical integrity* regarding food safety is detecting and classifying contaminants, that is, being able to determine what the contaminant is. As we shall show in this chapter, Hyperspectral X-ray analysis is very suited to this task, due to its capability to detect low density contaminants and discriminate them.

In this chapter we consider the problem of joint denoising and analysis of hyperspectral data acquired through a X-ray acquisition system. Given a single pixel of the used linear sensor, we propose a Convolutional Neural Network (CNN) that returns an estimate of a noise-free acquisition and detects the material scanned by the X-ray beam. The actual detection is done by means of estimation of two physical quantities related to the material, as explained in Section 5.3. First approaches to the use of CNNs for hyperspectral X-ray image denoising were proposed in [28, 213]. We now exploit what we learned from Chapter 6 to ease the classification process. Unlike the previously proposed approaches, our method works on single pixels, rather than complete linear acquisitions, making the approach much more flexible and capable of detecting smaller contaminants. Once the method is proved capable of identifying the presence of a polymer even in a single pixel, it can be used to analyze food products passing over a conveyor belt.

The chapter is organized as follows: Section 7.1 describes the proposed method for plastic polymers classification. Section 7.2 describes the experimental campaign we designed for validating our method. Section 7.3 provides an overview on the experimental results. Finally, Section 7.4 draws conclusions on the chapter.

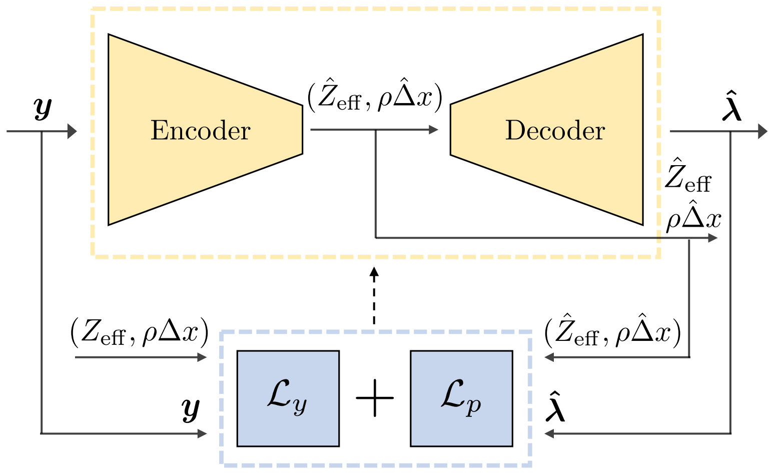

$\boldsymbol{y}$
Encoder
$(\hat{Z}_{\text{eff}}, \rho\hat{\Delta}x)$
Decoder
$\hat{\boldsymbol{\lambda}}$
$\hat{Z}_{\text{eff}}$
$\rho\hat{\Delta}x$
$(Z_{\text{eff}}, \rho\Delta x)$
$(\hat{Z}_{\text{eff}}, \rho\hat{\Delta}x)$
$\boldsymbol{y}$
$\mathcal{L}_y$
$+$
$\mathcal{L}_p$
$\hat{\boldsymbol{\lambda}}$

Figure 7.1: *Proposed network architecture and interaction between the CNN (yellow, top) and loss terms (blue, bottom).*

7.1 Proposed Method

The proposed technique is based on a CNN designed to achieve accurate performances while satisfying two application constraints: (i) the analysis must run in real-time, thus the CNN must be fast enough when compared to the system acquisition rate (i.e., overly complicated architectures are not a choice); (ii) materials must be correctly recognized independently from their size as even tiny objects might be harmful in food safety. We test the proposed system over a dataset of 347 classes, obtained by combining 13 materials (polymers and metals) with different thicknesses. The experimental campaign is conducted to test the system performance even when some polymers have never been seen by the CNN during training. Results show that it is possible to estimate polymers' parameters even by using a single pixel of the studied sensor. Moreover, results suggest that the proposed CNN opens the doors to the possibility of synthetic data generation, which might be useful whenever acquisitions should be simulated.

In this section, we provide the formal problem formulation and the details about the used architecture and the proposed custom loss function.

7.1.1 Problem formulation

The problem we consider is twofold. Given one noisy spectral acquisition $\boldsymbol{y}$ we are interested in: (i) estimate $\boldsymbol{\lambda}$, the clean version of it; (ii) estimate the two parameters Z_{eff} and $\rho\Delta x$ associated with the material. We consider the two estimations to be performed jointly as in a multi-task learning problem. Therefore, our goal is to design an operator $M(\cdot)$ such that:

$$\hat{\boldsymbol{\lambda}}, (\hat{Z}_{\text{eff}}, \rho\hat{\Delta}x) = M(\boldsymbol{y}), \tag{7.1}$$

where $\hat{\boldsymbol{\lambda}}$, $\hat{Z}_{\text{eff}}$ and $\rho\hat{\Delta}x$ represents the estimations of $\boldsymbol{\lambda}$, Z_{eff} and $\rho\Delta x$, respectively.

7.1.2 Network architecture

The proposed network architecture is depicted in Figure 7.1. Following the classical Convolutional AutoEncoder (CAE) paradigm, it is composed of an Encoder and a Decoder. The input is the spectral acquisition $\mathbf{y}$, whose dimensionality of B is reduced by the Encoder up to 2. The Encoder comprises 7 1D Convolutional Layers, each one followed by an ELU non-linearity layer [214]. Each layer's kernel size is used to reduce the signal's dimensionality through valid convolution, starting from a kernel size of 128 and halving it at each layer. The Decoder is designed with 7 1D Transposed Convolutional layers followed by ELUs to be symmetric to the Encoder and recover the original dimensionality B.

7.1.3 Loss function

The network is trained to minimise a compound loss function that takes into account two different kinds of errors: (i) the loss term $\mathscr{L}_y$ introduces a data fidelity term between the noisy input spectrum $\boldsymbol{y}$ and the reconstructed version of it $\hat{\boldsymbol{\lambda}}$; (ii) the loss term $\mathscr{L}_p$ takes the error on $\hat{Z}_{\text{eff}}$ and $\rho\hat{\Delta}x$ estimates into account. The generic form of the loss function is, therefore:

$$\mathscr{L} = \gamma\mathscr{L}_y + \mathscr{L}_p, \tag{7.2}$$

where γ is a weighting factor balancing the effect of one loss term over another.

We experimentally verified that the photons' count noise on each energy bin follows an independent Poissonian distribution. This allow us to employ a Weighted Log-Likelihood (WLL) as a data fidelity term $\mathscr{L}_y$:

$$\mathscr{L}_y = \frac{1}{B}\sum_{b=1}^{B}(\ln(P(y_b \mid \hat{\lambda}_b)) - \ln(P(y_b \mid y_b))), \tag{7.3}$$

where $P(y_b \mid \hat{\lambda}_b)$ represents the Probability Mass Function (pmf) for a Poisson distribution:

$$P(y_b \mid \hat{\lambda}_b) = \frac{\hat{\lambda}_b^{y_b} e^{-\hat{\lambda}_b}}{y_b!}. \tag{7.4}$$

and $P(y_b \mid y_b)$ is defined accordingly. By exploiting this prior, we do not need to know the actual clean ideal spectrum $\boldsymbol{\lambda}$ during training.

We also investigate the use of Mean Squared Error (MSE) as a term $\mathscr{L}_y$:

$$\mathscr{L}_y = \frac{1}{B}\sum_{b=1}^{B}\left(y_b - \hat{\lambda}_b\right)^2. \tag{7.5}$$

In this formulation, we are forcing the network to estimate the signal's clean version without explicitly inserting it in the equation. This is inspired by recent works on image denoising [211, 212], where the clean version of the input is unknown at training time.

As far as the term $\mathscr{L}_y$ is concerned, we exploit the Mean Absolute Percentage Error (MAPE) defined as

$$\mathscr{L}_p = \left|\frac{\hat{Z}_{\text{eff}}}{Z_{\text{eff}}} - 1\right| + \left|\frac{\rho\hat{\Delta}x}{\rho\Delta x} - 1\right|. \tag{7.6}$$

Notice that previous loss equations are defined for the i-th sample of the dataset. The global cost function on the whole dataset of N samples is defined as:

$$\mathscr{L}_{tot} = \frac{1}{N}\sum_{i}^{N}\mathscr{L}_i. \tag{7.7}$$

7.1.4 Deployment

Once the network is trained, we can feed it with a spectrum $\boldsymbol{y}$ and obtain the clean version of the acquisition $\hat{\boldsymbol{\lambda}}$ alongside the estimation of the two parameters $\hat{Z}_{\text{eff}}$ and $\rho\hat{\Delta}x$ related to the material. We can also consider the Encoder and the Decoder part of the CNN separately. Using only the Encoder will allow us to estimate the parameters $\hat{Z}_{\text{eff}}$ and $\rho\hat{\Delta}x$ for detection purposes, whereas the Decoder could be used as a synthetic generator of spectra given two arbitrary parameters as input.

7.2 Experiments

This section describes the performed experimental campaign.

7.2.1 Setup

To identify which combination of loss terms works best in the multi-task learning configuration and to study the impact of the different loss terms separately, we devised a series of experiments with different network configurations.

To validate the multi-task pipeline, we consider two scenarios:

A) Multitask AutoEncoder (AE), with WLL as $\mathscr{L}_y$ and MAPE as $\mathscr{L}_p$.

B) Multitask AE, with MSE as $\mathscr{L}_y$ and MAPE as $\mathscr{L}_p$.

In these two methods, the network input is the noisy spectral acquisition $\boldsymbol{y}$. The network outputs are the estimated clean spectrum $\hat{\boldsymbol{\lambda}}$ and the estimated parameters $\hat{Z}_{\text{eff}}$ and $\rho\hat{\Delta}x$ from the latent space of the network.

To study the effect of the single loss terms, we consider three scenarios:

C) Only Encoder, with MAPE as $\mathscr{L}_p$.

D) Full AE, with WLL as $\mathscr{L}_y$.

E) Full AE, with MSE as $\mathscr{L}_y$.

In scenario C, we consider only the encoding part of the network. The input is the noisy spectral acquisition $\boldsymbol{y}$, while the outputs are directly the latent space parameters $\hat{Z}_{\text{eff}}$ and $\rho\hat{\Delta}x$. In methods D and E, we do not consider the multi-task learning part, feeding the network $\boldsymbol{y}$ and obtaining $\hat{\boldsymbol{\lambda}}$ without extracting the latent space parameters.

We obtain $\boldsymbol{\lambda}$ by averaging $2\,560$ acquisition together. It is worth noting that excluding method C, we train all the other methods to reconstruct $\hat{\boldsymbol{\lambda}}$ (the mean version of the spectrum). However, the loss function $\mathscr{L}_y$ takes as input the noisy acquisition $\boldsymbol{y}$, and we never use $\boldsymbol{\lambda}$ during training. This choice is strictly related to the exploitation of the inherent prior introduced by $\mathscr{L}_y$, which allows the network to estimate the mean spectrum $\boldsymbol{\lambda}$ even without seeing one.

At test time, we are interested in measuring network performance. Therefore, we derive two metrics based on the adopted loss functions. In case of single parameters Z_{eff} and $\rho\Delta x$, we simply use $1-$MAPE to obtain a fidelity index $\mathrm{F}_p \in (-\infty, 1]$, the

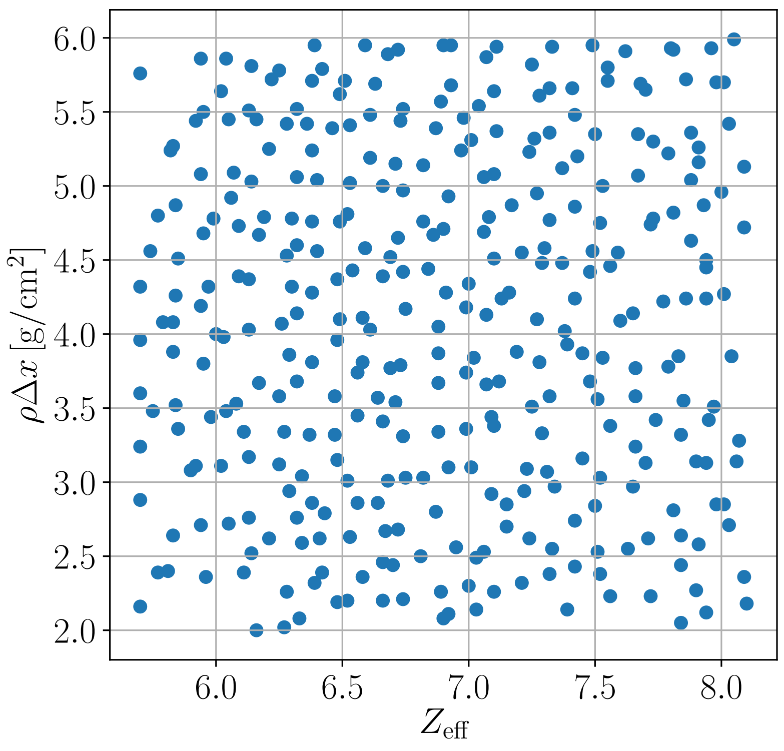


Figure 7.2: *Z_{eff} and $\rho\Delta x$ true distributions observed in the adopted dataset.*

higher the better. In the case of spectrum, we derive a compatibility index $\mathrm{A}_y \in [0, 1]$ by computing:

$$J = -\frac{1}{B}\sum_{b=1}^{B}(\ln(P(\lambda_b \mid \hat{\lambda}_b)) - \ln(P(\lambda_b \mid \lambda_b))), \tag{7.8}$$

and:

$$\mathrm{A}_y = \mathrm{e}^{-J}, \tag{7.9}$$

that is computing Equation (7.3) using the clean version of the spectrum $\boldsymbol{\lambda}$ instead of the noisy acquisition $\boldsymbol{y}$ and taking the negative exponential. When equal to 1, A_y indicates a perfect match between the two spectra, when equal to 0 it indicates a miserable match.

Regarding the implementation, we resort to Pytorch framework for developing the CNN. All the experiments are run on a workstation equipped with an NVIDIA Titan V Graphics Processing Unit (GPU), an Intel Xeon E5-2687W and 256 GiB RAM.

In addition to the polymers, acquisition of Aluminium tiles are present in the dataset, as it is a very common material in industrial machinery. Acquisitions are made both on single materials and on couples of materials, leading to a total of 347 different combinations of materials (e.g. tiling of different materials with different thicknesses). The distributions of Z_{eff} and $\rho\Delta x$ in the dataset are roughly uniform, as shown in Figure 7.2.

7.2.2 Dataset

The dataset is composed of acquisitions taken with the system mentioned above. The operating point of the X-ray tube is 60keV and 0.3 mA. We place each material object

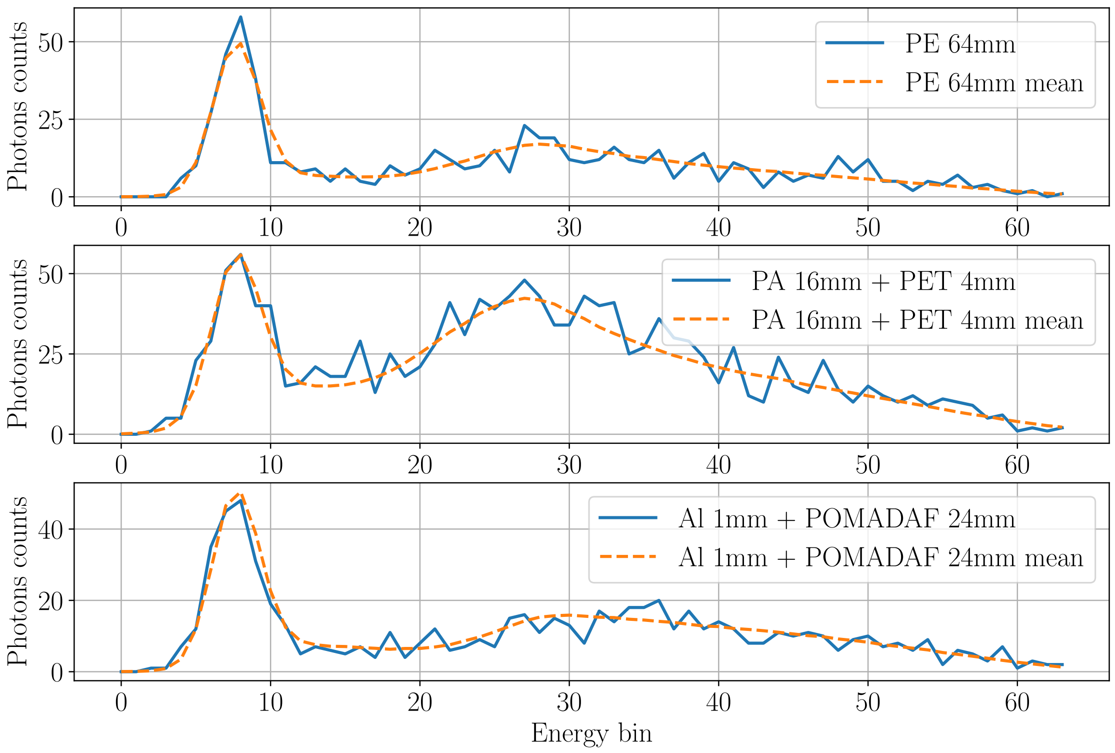


Figure 7.3: *Sample acquisitions from three different classes of the considered dataset. Blue curves show the actual noisy acquisition, whereas orange dashed curves show the clean ideal ground truth. Energy bins higher than $b = 64$ has been hidden for the sake of readability.*

in front of the sensor and irradiate it with an X-ray beam, recording the linear sensor output. For each object, we acquire 2 560 static images, with acquisition time 2ms (i.e., the active recording time of the sensor). The frequency spectrum of the acquisition is divided into $B = 256$ energy bins. We consider only the central pixel of the sensor. Therefore, each acquisition's final shape is (1×256) where the first dimension represents the pixel, and the second dimension is the number of energy bins used to represent the spectrum. Figure 7.3 shows three sample acquisitions, each belonging to a different class (i.e., combinations of material and thickness).

We consider 12 different polymers: 'PE', 'PA', 'PC', 'PMMA', 'PET', 'POM', 'POMADAF', 'PVDF', 'PTFE', 'PBT', 'PPS', 'PVC' and the metal Aluminium. These represent some of the most common contaminants that can be found during food inspection. Acquisitions are made both on single materials and on pairs of tiled materials. Considering all combinations of materials and thicknesses, we end up with 347 different classes, with a roughly uniform distribution of parameters Z_{eff} and $\rho\Delta x$, as shown in Figure 7.2.

The whole corpus of images is composed of more than $908\,445$ acquisitions of shape (1×256). We remove all the acquisitions of materials 'PA' and 'POMADAF' from the whole corpus. We also remove all the acquisitions of all the other materials at thickness 24mm and 36mm. From this reduced corpus, we use the 70% to create the train set $\mathbf{D}$, and the remaining 30% composes the first test set $\mathcal{T}$. We use an additional 20% of the train set as a validation set to evaluate the training procedure. The acquisitions of materials 'PA' and 'POMADAF' compose the second test set $\mathcal{T}_m$, while the acqui-

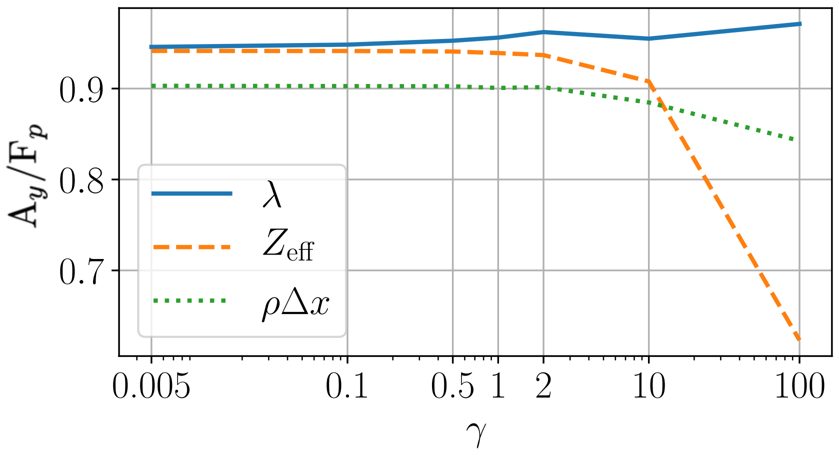


Figure 7.4: *Performances of* $\hat{\boldsymbol{\lambda}}$, $\hat{Z}_{\text{eff}}$ *and* $\rho\hat{\Delta x}$ *varying* γ.

sitions of all the materials with thickness 24mm and 36mm compose the third test set $\mathcal{T}_t$. The purpose of the last two test sets is to study the system behaviour with unknown (i.e., previously unseen) classes (i.e., materials and thicknesses). Those class pairs are randomly chosen.

7.3 Results

In this section we provide the result of the experimental campaign.

7.3.1 Parameter search

Since the loss function is a weighted average between two terms, we are interested in finding the best weighting factor γ. Therefore, we run a preliminary set of experiments on $\mathcal{T}$ varying γ. Results reported in Fig. Figure 7.4 shows how incrementing γ gives a little benefit in estimating $\hat{\boldsymbol{\lambda}}$, while affecting the estimation of $\hat{Z}_{\text{eff}}$ and $\rho\hat{\Delta x}$. To obtain a balance between the three estimations, we fixed $\gamma = 0.1$.

7.3.2 Results on known dataset

We report results for the known dataset $\mathcal{T}$ in Table 7.1. Regarding $\boldsymbol{\lambda}$, the multitask approach A offers a better performance than its single task counterpart E, while B is comparable with D. The performance on Z_{eff} and $\rho\Delta x$ is pretty much the same when comparing with method C. Probably, the estimation of the two parameters is a more straightforward task for the network, and it does not need any help from the multitask learning paradigm. Conversely, the clean signal estimation is a more challenging task and benefits from the joint loss function.

7.3.3 Results on unknown datasets

Table 7.2 and Table 7.3 show the results when we test on unknown materials $\mathcal{T}_m$ and thicknesses $\mathcal{T}_t$. We can confirm the trend of Table 7.1: in both cases, the network reaches the best accuracy on estimating $\boldsymbol{\lambda}$ when training in a multitask fashion. The estimation of Z_{eff} and $\rho\Delta x$ seems to suffer less from the lack of multitasking.

Table 7.1: *Results on known dataset $\mathcal{T}$.*

Method	Z_{eff} [F$_p$]	$\rho\Delta x$ [F$_p$]	$\boldsymbol{\lambda}$ [A$_y$]
MAPE + MSE (multitask) (A)	0.94227	0.90416	0.97946
MAPE + Poisson (multitask) (B)	0.93868	0.90469	0.96163
MAPE ($\rho\Delta x$, Z_{eff}) (C)	0.94140	0.90355	–
MSE ($\boldsymbol{\lambda}$) (E)	–	–	0.93835
Poisson ($\boldsymbol{\lambda}$) (D)	–	–	0.96395

Table 7.2: *Results on unknown dataset $\mathcal{T}_m$ (unknown materials).*

Method	Z_{eff} [F$_p$]	$\rho\Delta x$ [F$_p$]	$\boldsymbol{\lambda}$ [A$_y$]
MAPE + MSE (multitask)	0.93758	0.90864	0.97864
MAPE + Poisson (multitask)	0.93462	0.90947	0.95950
MAPE ($\rho\Delta x$, Z_{eff})	0.93659	0.90817	–
MSE ($\boldsymbol{\lambda}$)	–	–	0.93613
Poisson ($\boldsymbol{\lambda}$)	–	–	0.96267

Table 7.3: *Results on unknown dataset $\mathcal{T}_t$ (unknown thicknesses).*

Method	Z_{eff} [F$_p$]	$\rho\Delta x$ [F$_p$]	$\boldsymbol{\lambda}$ [A$_y$]
MAPE + MSE (multitask)	0.94062	0.90428	0.97930
MAPE + Poisson (multitask)	0.93676	0.90276	0.96203
MAPE ($\rho\Delta x$, Z_{eff})	0.93976	0.90394	–
MSE ($\boldsymbol{\lambda}$)	–	–	0.93288
Poisson ($\boldsymbol{\lambda}$)	–	–	0.96506

7.4 Conclusions

In this chapter we proposed a method for jointly denoising a hyperspectral X-ray acquisition and estimating the two physical parameters related to the acquired object. We tested our approach to a vast dataset of polymers and metal acquisitions, considering the case in which the network has never seen the material under analysis. We compared the multitask paradigm with the separate estimation of the two desired outputs (e.g., the parameters and the mean spectrum), showing benefits in adopting the former. The results show that it is possible to employ multitask learning to develop a polymer classification pipeline tailored to industrial food supply chain. As a future work, we will take advantage of the Encoding/Decoding nature of the considered CAE to use it as a physical generator of synthetic data. We can in fact apply just the decoding part of the network to retrieve plausible estimation of a real spectrum by feeding the network with a given couple of Z_{eff} and $\rho\Delta x$.

CHAPTER 8

Conclusions

This thesis investigates the integrity control problem focusing on two specific applications: Multimedia Forensics for the *digital integrity* part and Hyperspectral X-ray analysis in food control for the *physical integrity* part.

Chapter 2 treats the sensor integrity, that is, the integrity of the traces left by the camera sensor in the image. In particular, we focus on the well known Photo Response Non Uniformity (PRNU), a unique noise pattern that allows binding an image to the device that shot it. We propose a method for anonymizing an image, i.e., removing the PRNU traces, being careful to retain the visual quality. The idea behind the method is to treat a Convolutional Neural Network (CNN) as a parametric operator, to be trained specifically on each image to anonymize. To this extent, instead of training the network to minimize a loss function over a large dataset of images, we explicitly want the network to overfit every single image.

We also consider the opposite problem, which is image anonymization detection. We propose a method for detecting image anonymization based on a feature vector extracted from the RGB domain and then fed to a CNN. This approach mixes model-based feature extraction and data-driven learning approach, indicating that one world can benefit from another. The results show that training the network on the extracted feature vector performs better than training the network directly on the image (pure data-driven approach) in the cross-dataset testing scenario. Moreover, the results indicate that our method can generalize well in very arduous testing conditions. It is the best performer (with respect to the other state-of-the-art method) over unseen devices and is robust against a common set of non-geometric transformations (e.g., JPEG compression, gamma correction, brightness correction, contrast correction). The investigation of the performance degradation after post-processing operations is one of the added values of this proposed method. Such a test is rarely found in other state-state-of-the-art methods, while it is paramount to build effective methods that can be applied in real-case scenarios.

Chapter 3 is about the integrity of the coding traces left by multiple JPEG compressions, which can be a hint of tampering with the image. We investigate the double JPEG compression detection problem, dealing with both the possibilities of an aligned and a non-aligned second compression with respect to the first one's 8×8 grid. We propose a Deep Learning method based on a CNN, considering different domains as input. To the best of our knowledge, this is the first method in the literature to consider the problem of double JPEG detection in a data-driven fashion. Results show how the proposed method can detect the double JPEG compression in both the considered scenarios, outperforming the baselines even in the complex case in which the Quality Factor (QF) of the second compression is greater than the first one.

Extending our research, we also investigate the multiple JPEG compression detection problem. Images are often uploaded and downloaded several times from various social media. Therefore, detecting a higher number of consecutive compressions is paramount for forensics analysts. We propose a method for detecting multiple JPEG compressions up to four coding steps. This method employs the Task-driven Non-negative Matrix Factorization (TNMF) model, which jointly allows learning a feature from data and iteratively performing the actual classification. The results show that the proposed solution is better than the state-of-the-art baseline for classification accuracy.

Finally, we consider the problem of detecting different JPEG implementations in double compression. This is an essential part of the whole problem of multiple JPEG compression detection since the JPEG standard does not define strict rules for the quantization of the Discrete Cosine Transform (DCT) coefficients of the image. Everyone can implement their lossy part of the compression algorithm, meaning that we can look for traces left by every different implementation. We propose a novel method based on the eigen-algorithms idea. This method takes advantage of recompressing the same image multiple times with a set of different JPEG implementations and then exploiting the deviations introduced by their diversity as a discriminative feature. We tested different use cases, including software detection and the detection of a JPEG anti-forensic technique. The results show that we can achieve high accuracy in different scenarios.

Chapter 4 discusses the semantic integrity of images and videos. In particular, we care about assessing the generation process of an image or a video. The ultimate goal is to tell real content (i.e., acquired with a digital camera) and generated content (i.e., synthetically produced by neural networks) apart. The recent spreading of easy to use generation methods can lead to an outbreak of malicious usages, to name a few, false impersonation and puppeteering. Regarding images, we focus on detecting generation by Generative Adversarial Networks (GANs), the most common and powerful technique for still image generation nowadays. Rather than proposing a purely data-driven approach, we consider the statistical properties of image generation. The proposed method exploits Benford's law to extract feature vectors out of the DCT coefficients of an image. We test multiple setups in which different combinations of the features are combined and fed to a Random Forest classifier to discriminate the fake images from the real ones. The results show how the proposed feature extraction method is powerful enough to work even if paired with a simple classifier like the Random Forest. We can conclude that GAN generated images struggle in respecting Benford's law, even if some of the latest architectures are more difficult to detect. The simplicity of this solution makes it suitable in cases in which the computational budget is a concern, e.g., mobile phones, embedded devices.

Regarding videos, we investigated the recent trend of Deepfakes, a generative method born with the intent of substituting a person's face with someone else's face. We propose a method that takes advantage of the concept of model ensembling. A set of CNNs is used in the inference phase, instead of just one network, eventually aggregating the scores to provide the final classification label. We modify and train different versions of a backbone network to exploit two main concepts: (i) an attention mechanism focusing on certain facial regions; (ii) a triplet siamese training strategy aimed to ease the training procedure in an extensive dataset of videos. We evaluate the procedure over two publicly available datasets containing almost $120\,000$ videos, showing the effectiveness of our method. We also participated in a coding challenge tailored to this problem and sponsored by Facebook. The proposed solution ranked in the top 2% over more than 2000 participants.

Chapter 5 introduces the second part of the thesis, i.e., the part related to *physical integrity*. In this chapter we introduce the reader to the problem of integrity estimation in the food industry, explaining how non-destructive methods are preferable over intrusive ones. We then introduce the X-ray acquisition system used in our study. This is a commonly used system for food control. It is composed by a X-ray generator, a conveyor belt transporting the objects under analysis (e.g., food packages) and a Hyperspectral X-ray detector. The detector is a linear sensor where each cell is capable of counting impinging photons separating different energy levels. The broad energy range this system is able to capture allows to perform very fine analysis on the acquired objects, detecting even low-density contaminants. Finally, we introduce Lambert-Beer law, which is the physical law we exploit in the coming chapters to link a physical material to be detected, with the X-ray acquired data.

Chapter 6 faces the problem of denoising Hyperspectral X-ray acquisitions, a crucial step for developing further processing steps such as classification. In this vein, we present two original contributions. In the first one, we compare a model-based method for denoising, the Wiener filter, with a data-driven technique, a Convolutional AutoEncoder (CAE). Experiments on plastic polymers acquisitions show that both methods can reach good denoising capabilities. However, the CAE has the advantage of not depending on the tedious estimation of the Noise-to-Signal ratio. Moreover, being the CAE a data-driven model, we can exploit transfer learning and domain adaptation to use it as a starting point for other networks dedicated to other tasks. Once we assess that the task of Hyperspectral X-ray denoising could be done through Deep Learning techniques, we proceed with the second contribution, a comparison between different AutoEncoder (AE) variants. In fact, AEs has proved a very efficient architecture for denoising purposes in the literature. In our experiments, we compare the basic AE with the Variational AutoEncoder (VAE) and a deterministic variant of the latter we call Augmented AutoEncoder (AuAE). We force the latent space of those models to be as little as two parameters, a stringent condition motivated by the physical law of Lambert-Beer. Our experiments show how spectra acquired from plastic polymers and "low-density" metals can be polished from noise in an unsupervised fashion, i.e., not having access to the "clean" reference spectrum. We validate our method on extensive synthetic and real acquisition datasets. All the networks show excellent results despite the reduced latent space. Nonetheless, we can conclude that the AuAE is the best option for real-time applications given the performances and its deterministic behaviour at inference time.

Chapter 7 investigates the task of low-density polymer classification starting from Hyperspectral X-ray acquisitions. This is a very interesting task in the considered food safety framework, as knowing the composition of a detected contaminant is as important as detecting it. To solve this problem, we propose a network for estimating two physical parameters associated with a given material starting from its X-ray spectrum. The reason behind this simplified choice is that, once we can determine the contaminant from pure contaminant's acquisition, we can learn a "signature" of the contaminant and then adjust the estimation for the case the contaminant is buried in packaged food. We exploit the multitask paradigm to train our network. We jointly perform the denoising of the acquisition while estimating the parameters. In this way, we can correctly classify the inspected polymer while obtaining a clean version of the acquisition that we can use for further processing. Results show how the multitask paradigm benefits the method, outperforming two networks that execute the denoising and the classification tasks separately. Besides, the trained network can also be used afterwards as a synthetic generator of spectra. In fact, by using only the decoding part of the network, one can easily retrieve a synthetic estimation of the spectrum associated with the pair of physical input parameters associated with the Lambert-Beer law.

8.1 Future research

By investigating the problem of integrity assessment at digital and physical level, we notice how some problems still have to be addressed. In this section we offer possible future advancements for the proposed methods, considering the possibilities and the innovations of the state of the art in the near future. At the same time, we would like to outline future research trends in the respective fields.

Regarding sensor integrity, it would be fascinating to embed the state-of-the-art Wavelet-based denoising function used in the PRNU testing directly into the network. This is not feasible with the current Deep Learning frameworks, as the Wavelet function is not differentiable. This breaks the backpropagation paradigm used by neural networks for the weights update. Recent Deep Learning library such Pytorch [166] and TensorFlow [215] have made giant leaps in offering implementation possibilities with respect to the limitations frameworks had just five years ago. We hope that this enhancements will continue in the future, making the proposed improvement possible. This would dramatically increase the impact of the proposed method, as even if analysts knew about the possible attack, it would be practically impossible for them to discover it through the correlation test.

Regarding coding integrity, we showed how a problem that has always been addressed by model-based solutions could benefit from the data-driven world of CNN. However, we should not forget how the pre-processing and the careful implementation of handcrafted features can benefit the pure data-driven approach. For the future, it would be exciting to investigate the mixed approach (model-based and data-driven) adopted for the double JPEG problem also for the task of multiple JPEG compression detection.

Semantic integrity offers the most thrilling future scenario in the Multimedia Forensics panorama. We showed how we can effectively detect generated content, but Pandora's box has just been opened. Soon, we will see more and more fake images and videos, and as time passes, they will become even more indistinguishable to the human eye. Unfortunately, every pure data-driven approach for the detection will always be a step behind the generators, as the typical pipeline for detecting generated content requires a certain amount of data from the specific generative architecture. For this reason, it is of vital importance to keep working on mixed approaches in which the statistical properties of images and videos are considered as a starting point to implement better detection methods.

Regarding the study of denoising for physical integrity, it would be of great interest to investigate the effect of reducing the latent space of the network even more. Preliminary results show a certain degree of correlation between the two learned parameters of the latent space. This suggests that a possible parameterization of the materials made with just one parameter could exist, which would significantly impact the number of parameters of the network, its performances, and the inference time.

Regarding classification of polymers, the natural continuation would be to investigate the case in which we take the acquisitions from food packages containing the contaminants. In this case, it would be interesting to assess if the network can learn to exclude the "common" area of the acquisition representing the food, acting as an anomaly detector when encountering a contaminant. More realistically, an extensive training campaign providing acquisitions of both "virgin" and contaminated packages is required for the network to learn an appropriate discriminative feature.

Bibliography

[1] D. Beaver, S. Kumar, H. C. Li, J. Sobel, P. Vajgel, *et al.*, “Finding a needle in haystack: Facebook’s photo storage,” *Proceedings of the USENIX Conference on Operating Systems Design and Implementation*, vol. 10, pp. 1–8, Oct. 2010. Vancouver, BC, Canada.

[2] S. Alhabash and M. Ma, “A tale of four platforms: Motivations and uses of facebook, twitter, instagram, and snapchat among college students?,” *Social Media + Society*, vol. 3, Feb. 2017.

[3] S. Bradshaw and P. N. Howard, “The global disinformation disorder: 2019 global inventory of organised social media manipulation,” *Oxford, UK: Project on Computational Propaganda*, 2019.

[4] S. Bradshaw and P. N. Howard, “Challenging truth and trust: A global inventory of organized social media manipulation,” *The Computational Propaganda Project*, 2018.

[5] S. C. Woolley and P. N. Howard, *Computational propaganda: political parties, politicians, and political manipulation on social media.* Oxford University Press, 2018.

[6] N. Marriott and R. Gravani, “The role of HACCP in sanitation,” in *Principles of Food Sanitation. Food Science Texts Series*, pp. 99–115, Springer, Boston, MA, 2006.

[7] S. Mathanker, P. Weckler, and T. Bowser, “X-ray applications in food and agriculture: a review,” *Transactions of the ASABE (American Society of Agricultural and Biological Engineers)*, vol. 56, pp. 1227–1239, 05 2013.

[8] M. Goljan and J. Fridrich, “Camera identification from cropped and scaled images,” in *Proceedings of the SPIE Electronic Imaging*, vol. 6819, pp. 6819–6819–13, January 2008. San Jose, CA.

[9] S. Mandelli, L. Bondi, S. Lameri, V. Lipari, P. Bestagini, and S. Tubaro, “Inpainting-based camera anonymization,” in *IEEE International Conference on Image Processing (ICIP)*, 2017.

[10] A. E. Dirik, H. T. Sencar, and N. 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, pp. 2277–2290, Dec. 2014.

[11] C. Chen, Y. Q. Shi, and W. Su, “A machine learning based scheme for double JPEG compression detection,” in *IEEE International Conference on Pattern Recognition (ICPR)*, 2008.

[12] T. Bianchi and A. Piva, “Detection of nonaligned double JPEG compression based on integer periodicity maps,” *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 7, pp. 842–848, 2012.

[13] Q. Wang and R. Zhang, “Double JPEG compression forensics based on a convolutional neural network,” *EURASIP Journal on Information Security*, vol. 2016, pp. 1–23, December 2016.

[14] C. Pasquini, G. Boato, and F. Perez-Gonzalez, “Multiple jpeg compression detection by means of benford-fourier coefficients,” in *Information Forensics and Security (WIFS), 2014 IEEE International Workshop on*, pp. 113–118, IEEE, 2014.

[15] W. B. Pennebaker and J. L. Mitchell, *JPEG Still Image Data Compression Standard.* New York: Van Nostrand Reinhold, 1992.

[16] L. Verdoliva, “Media forensics and deepfakes: an overview,” 2020.

[17] "Faceswap." `https://github.com/MarekKowalski/FaceSwap/`.

[18] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning, (ICML) 2019*, vol. 97 of *Proceedings of Machine Learning Research*, pp. 6105–6114, PMLR, 2019.

[19] F. W. Spiers, "Effective atomic number and energy absorption in tissues," *The British Journal of Radiology*, vol. 19, no. 218, pp. 52–63, 1946.

[20] G. Bubba, "Material identification and Compton scattering effects in X-ray multienergy radiography of homogeneous compounds," *POLITesi - Politecnico di Milano*, 2017.

[21] N. Bonettini, L. Bondi, D. Güera, S. Mandelli, P. Bestagini, S. Tubaro, and E. J. Delp, "Fooling prnu-based detectors through convolutional neural networks," *Proceedings of the IEEE European Signal Processing Conference*, Sept. 2018. Rome, Italy.

[22] N. Bonettini, D. Güera, L. Bondi, P. Bestagini, E. J. Delp, and S. Tubaro, "Image anonymization detection with deep handcrafted features," in *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 2304–2308, 2019.

[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] S. Mandelli, N. Bonettini, P. Bestagini, V. Lipari, and S. Tubaro, "Multiple jpeg compression detection through task-driven non-negative matrix factorization," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2106–2110, 2018.

[25] N. Bonettini, L. Bondi, P. Bestagini, and S. Tubaro, "Jpeg implementation forensics based on eigen-algorithms," in *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 1–7, 2018.

[26] N. Bonettini, P. Bestagini, S. Milani, and S. Tubaro, "On the use of benford's law to detect gan-generated images," in *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 5495–5502, 2021.

[27] N. Bonettini, E. D. Cannas, S. Mandelli, L. Bondi, P. Bestagini, and S. Tubaro, "Video face manipulation detection through ensemble of cnns," in *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 5012–5019, 2021.

[28] N. Bonettini, M. Paracchini, P. Bestagini, M. Marcon, and S. Tubaro, "Hyperspectral X-ray denoising: model-based and data-driven solutions," in *European Signal Processing Conference (EUSIPCO)*, 2019.

[29] N. Bonettini, C. Gonano, P. Bestagini, M. Marcon, B. Garavelli, and S. Tubaro, "Multitask learning for denoising and analysis of X-ray polymer acquisitions," in *European Signal Processing Conference (EUSIPCO)*, 2021.

[30] S. Mandelli, N. Bonettini, P. Bestagini, and S. Tubaro, "Training cnns in presence of jpeg compression: Multimedia forensics vs computer vision," in *2020 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 1–6, 2020.

[31] S. Mandelli, N. Bonettini, and P. Bestagini, "Source camera model identification," in *Multimedia Forensics* (H. T. Sencar, L. Verdoliva, and N. Memon, eds.), Advances in Computer Vision and Pattern Recognition, pp. 133–173, Springer Singapore, 2022.

[32] E. D. Cannas, N. Bonettini, S. Mandelli, P. Bestagini, and S. Tubaro, "Amplitude sar imagery splicing localization," *IEEE Access*, vol. 10, pp. 33882–33899, 2022.

[33] M. Kirchner and T. Gloe, "Forensic camera model identification," in *Handbook of Digital Forensics of Multimedia Data and Devices*, pp. 329–374, Chichester, UK: John Wiley & Sons, Ltd, 2015.

[34] S. Bayram, H. Sencar, N. Memon, and I. Avcibas, "Source camera identification based on CFA interpolation," *Proceedings of the IEEE International Conference on Image Processing*, pp. 69–72, September 2005. Genova, Italy.

[35] X. Zhao and M. C. Stamm, "Computationally efficient demosaicing filter estimation for forensic camera model identification," *Proceedings of the IEEE International Conference on Image Processing*, pp. 151–155, September 2016. Phoenix, AZ.

[36] S.-H. Chen and C.-T. Hsu, "Source camera identification based on camera gain histogram," *Proceedings of the IEEE International Conference on Image Processing*, pp. 429–432, September 2007. San Antonio, TX.

[37] C. Chen and M. C. Stamm, "Camera model identification framework using an ensemble of demosaicing features," *IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 1–6, November 2015. Rome, Italy.

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

[39] A. Tuama, F. Comby, and M. Chaumont, "Camera model identification with the use of deep convolutional neural networks," in *IEEE International Workshop on Information Forensics and Security (WIFS)*, 2016.

[40] L. Bondi, L. Baroffio, D. Güera, P. Bestagini, E. J. Delp, and S. Tubaro, "First steps toward camera model identification with convolutional neural networks," *IEEE Signal Processing Letters (SPL)*, vol. 24, pp. 259–263, March 2017.

[41] B. Bayar and M. C. Stamm, "Design principles of convolutional neural networks for multimedia forensics," in *IS&T International Symposium on Electronic Imaging: Media Watermarking, Security, and Forensics*, 2017.

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

[43] J. Lukáš, J. Fridrich, and M. Goljan, "Digital camera identification from sensor pattern noise," *IEEE Transactions on Information Forensics and Security*, vol. 1, pp. 205–214, June 2006.

[44] M. Kirchner and C. Johnson, "Spn-cnn: Boosting sensor-based source camera attribution with deep learning," in *2019 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 1–6, IEEE, 2019.

[45] S. Mandelli, D. Cozzolino, P. Bestagini, L. Verdoliva, and S. Tubaro, "Cnn-based fast source device identification," *IEEE Signal Processing Letters*, vol. 27, pp. 1285–1289, 2020.

[46] M. Goljan, M. Chen, P. Comesaña, and J. Fridrich, "Effect of compression on sensor-fingerprint based camera identification," *Proceedings of the IS&T Electronic Imaging*, vol. 2016, pp. 1–10, Jan. 2016. Burlingame, CA.

[47] P. C. Giannelli, "Daubert and forensic science: The pitfalls of law enforcement control of scientific research," *University of Illinois Law Review*, p. 53, 2011.

[48] M. Chen, J. Fridrich, M. Goljan, and J. Lukas, "Source digital camcorder identification using sensor photo-response nonuniformity," in *SPIE Electronic Imaging (EI)*, Mar. 2007. San Jose, CA.

[49] M. Chen, J. Fridrich, M. Goljan, and J. Lukáš, "Determining image origin and integrity using sensor noise," *IEEE Transactions on Information Forensics and Security*, vol. 3, pp. 74–90, Mar. 2008.

[50] J. Entrieri and M. Kirchner, "Patch-based desynchronization of digital camera sensor fingerprints," *Proceedings of the IS&T Electronic Imaging*, vol. 2016, pp. 1–9, Jan. 2016. Burlingame, CA.

[51] M. Brundage *et al.*, "The malicious use of artificial intelligence: Forecasting, prevention, and mitigation," *arXiv:1802.07228*, Feb. 2018.

[52] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE Transactions on Image Processing*, vol. 26, pp. 3142–3155, July 2017.

[53] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proceedings of the 30th International Conference on Machine Learning* (S. Dasgupta and D. McAllester, eds.), vol. 28 of *Proceedings of Machine Learning Research*, (Atlanta, Georgia, USA), pp. 1139–1147, PMLR, 17–19 Jun 2013.

[54] T. Gloe and R. Böhme, "The Dresden image database for benchmarking digital image forensics," *Journal of Digital Forensic Practice*, vol. 3, pp. 150–159, December 2010.

[55] M. K. Mihcak, I. Kozintsev, K. Ramchandran, and P. Moulin, "Low-complexity image denoising based on statistical modeling of wavelet coefficients," *IEEE Signal Processing Letters (SPL)*, vol. 6, pp. 300–303, 1999.

[56] 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.

[57] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, pp. 1798–1828, Aug 2013.

[58] T. Teng, A. Tan, and J. M. Zurada, "Self-organizing neural networks integrating domain knowledge and reinforcement learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, pp. 889–902, May 2015.

[59] K. Weiss, T. M. Khoshgoftaar, and D. Wang, “A survey of transfer learning,” *Journal of Big Data*, vol. 3, p. 9, May 2016.

[60] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *Proceedings of the International Conference on Learning Representations*, vol. arXiv, May 2015. San Diego, CA.

[61] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, June 2016. Las Vegas, NV.

[62] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Advances in Neural Information Processing Systems*, pp. 1097–1105, Dec. 2012. Lake Tahoe, NV.

[63] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *Proceedings of the International Conference on Learning Representations*, vol. arXiv, May 2015. San Diego, CA.

[64] M. Kirchner, “Fast and reliable resampling detection by spectral analysis of fixed linear predictor residue,” in *ACM workshop on Multimedia and security (MM&Sec)*, Sept. 2008. Oxford, UK.

[65] K. Rosenfeld and H. T. Sencar, “A study of the robustness of PRNU-based camera identification,” *Proceedings of the IS&T/SPIE Electronic Imaging*, Jan. 2009. San Jose, CA.

[66] A. Rocha, W. Scheirer, T. Boult, and S. Goldenstein, “Vision of the unseen: Current trends and challenges in digital image and video forensics,” *ACM Computing Surveys (CSUR)*, vol. 43, pp. 26:1–26:42, October 2011.

[67] A. Piva, “An overview on image forensics,” *ISRN Signal Processing*, vol. 2013, p. 22, Nov. 2013.

[68] A. Popescu and H. Farid, “Statistical tools for digital forensics,” in *International Conference on Information Hiding*, 2004.

[69] T. Pevny and J. Fridrich, “Detection of double-compression in jpeg images for applications in steganography,” *IEEE Transactions on Information Forensics and Security*, vol. 3, pp. 247–258, June 2008.

[70] B. Li, Y. Shi, and J. Huang, “Detecting doubly compressed JPEG images by using mode based first digit features,” in *IEEE Workshop on Multimedia Signal Processing (MMSP)*, 2008.

[71] P. Korus and J. 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, 2016.

[72] A. Taimori, F. Razzazi, A. Behrad, A. Ahmadi, and M. Babaie-Zadeh, “Quantization-unaware double jpeg compression detection,” *Journal of Mathematical Imaging and Vision*, vol. 54, no. 3, pp. 269–286, 2016.

[73] C. Pasquini, P. Schöttle, R. Böhme, G. Boato, and F. 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*, pp. 11–21, ACM, 2016.

[74] Z. Lin, J. He, X. Tang, and C. K. Tang, “Fast, automatic and fine-grained tampered JPEG image detection via DCT coefficient analysis,” *Pattern Recognition*, vol. 42, pp. 2492–2501, 2009.

[75] T. Bianchi, A. De Rosa, and A. Piva, “Improved DCT coefficient analysis for forgery localization in JPEG images,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2011.

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

[77] W. Luo, Z. Qu, J. Huang, and G. Qiu, “A novel method for detecting cropped and recompressed image block,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2007.

[78] Y.-L. Chen and C.-T. Hsu, “Detecting recompression of JPEG images via periodicity analysis of compression artifacts for tampering detection,” *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 6, pp. 396–406, 2011.

[79] Z. Qu, W. Luo, and J. Huang, “A convolutive mixing model for shifted double JPEG compression with application to passive image authentication,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2008.

[80] T. Bianchi and A. Piva, “Image forgery localization via block-grained analysis of jpeg artifacts,” *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 7, pp. 1003–1017, 2012.

[81] V. Conotter, P. Comesana, and F. Perez-Gonzalez, “Forensic analysis of full-frame linearly filtered JPEG images,” in *IEEE International Conference on Image Processing (ICIP)*, 2013.

[82] C. Pasquini, G. Boato, and F. Pérez-González, “Multiple JPEG compression detection by means of Benford-Fourier coefficients,” in *IEEE International Workshop on Information Forensics and Security (WIFS)*, 2014.

[83] F. Pérez-González, G. L. Heileman, and C. T. Abdallah, “Benford’s lawin image processing,” in *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, vol. 1, pp. I–405, IEEE, 2007.

[84] S. Milani, M. Tagliasacchi, and S. Tubaro, “Discriminating Multiple JPEG Compressions Using First Digit Features,” *APSIPA Transactions on Signal and Information Processing*, vol. 3, no. May, pp. 1–11, 2014.

[85] S. Agarwal and H. Farid, “Photo forensics from JPEG dimples,” in *IEEE International Workshop on Information Forensics and Security (WIFS)*, 2017.

[86] R. C. Gonzalez and R. E. Woods, “Image processing,” *Digital image processing*, vol. 2, 2007.

[87] Y. Qian, J. Dong, W. Wang, and T. Tan, “Deep learning for steganalysis via convolutional neural networks,” in *SPIE Media Watermarking, Security, and Forensics*, International Society for Optics and Photonics, 2015.

[88] L. Pibre, P. Jérôme, D. Ienco, and M. Chaumont, “Deep learning for steganalysis is better than a rich model with an ensemble classifier, and is natively robust to the cover source-mismatch,” in *IS&T International Symposium on Electronic Imaging: Media Watermarking, Security, and Forensics*, no. 8, February 2016. San Francisco, CA.

[89] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, pp. 2278–2323, 1998.

[90] J. Chen, X. Kang, Y. Liu, and Z. J. Wang, “Median Filtering Forensics Based on Convolutional Neural Networks,” *IEEE Signal Processing Letters*, vol. 22, pp. 1849–1853, November 2015.

[91] B. Bayar and M. C. Stamm, “A deep learning approach to universal image manipulation detection using a new convolutional layer,” *ACM Workshop on Information Hiding and Multimedia Security (IHMMSEC)*, 2016.

[92] D.-T. Dang-Nguyen, C. Pasquini, V. Conotter, and G. Boato, “RAISE: A raw images dataset for digital image forensics,” in *ACM Multimedia Systems Conference*, 2015.

[93] J. Yang, J. Xie, G. Zhu, S. Kwong, and Y.-Q. 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, 2014.

[94] R. Serizel, V. Bisot, S. Essid, and G. Richard, “Supervised group nonnegative matrix factorisation with similarity constraints and applications to speaker identification,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2017.

[95] J. Mairal, F. Bach, and J. Ponce, “Task-driven dictionary learning,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, pp. 791–804, 2012.

[96] V. Bisot, R. Serizel, S. Essid, and G. Richard, “Supervised nonnegative matrix factorization for acoustic scene classification,” *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2016.

[97] C. M. Bishop, “Pattern recognition,” *Machine Learning*, vol. 128, pp. 209–210, 2006.

[98] J. Nocedal, “Updating quasi-newton matrices with limited storage,” *Mathematics of computation*, vol. 35, no. 151, pp. 773–782, 1980.

[99] G. Schaefer and M. Stich, “UCID: an uncompressed color image database,” in *Storage and Retrieval Methods and Applications for Multimedia 2004* (M. M. Yeung, R. W. Lienhart, and C.-S. Li, eds.), vol. 5307, pp. 472–480, 2003.

[100] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, “Online learning for matrix factorization and sparse coding,” *Journal of Machine Learning Research*, vol. 11, no. Jan, pp. 19–60, 2010.

[101] S. Milani, P. Bestagini, M. Tagliasacchi, and S. Tubaro, “Demosaicing strategy identification via eigenalgorithms,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2659–2663, May 2014.

[102] F. Huang, J. Huang, and Y. Q. Shi, “Detecting double JPEG compression with the same quantization matrix,” *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 5, pp. 848–856, 2010.

[103] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[104] G. Schaefer and M. Stich, “UCID: An uncompressed colour image database,” in *SPIE Storage and Retrieval Methods and Applications for Multimedia*, 2004.

[105] NIST, "Nimble Challenge 2017 Dataset (NC2017) for image manipulation detection." Available at: https://www.nist.gov/itl/iad/mig/media-forensics-challenge.

[106] M. Barni, E. Nowroozi, and B. Tondi, "Higher-order, adversary-aware, double jpeg-detection via selected training on attacked samples," in *European Signal Processing Conference (EUSIPCO), to be presented at*, pp. 113–118, IEEE, 2017.

[107] A. Bansal, S. Ma, D. Ramanan, and Y. Sheikh, "Recycle-GAN: Unsupervised video retargeting," in *European Conference on Computer Vision (ECCV)*, 2018.

[108] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[109] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[110] F. Marra, D. Gragnaniello, D. Cozzolino, and L. Verdoliva, "Detection of GAN-Generated Fake Images over Social Networks," *IEEE International Conference on Multimedia Information Processing and Retrieval (MIPR)*, 2018.

[111] F. Marra, D. Gragnaniello, L. Verdoliva, and G. Poggi, "Do GANs Leave Artificial Fingerprints?," *IEEE International Conference on Multimedia Information Processing and Retrieval (MIPR)*, 2019.

[112] S. McCloskey and M. Albright, "Detecting GAN-generated imagery using saturation cues," in *IEEE International Conference on Image Processing (ICIP)*, 2019.

[113] H. Farid and M. J. Bravo, "Perceptual discrimination of computer generated and photographic faces," *Digital Investigation*, vol. 8, pp. 226–235, 2012.

[114] D. Dang-Nguyen, G. Boato, and F. G. B. De Natale, "3d-model-based video analysis for computer generated faces identification," *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 10, pp. 1752–1763, 2015.

[115] N. Rahmouni, V. Nozick, J. Yamagishi, and I. Echizen, "Distinguishing computer graphics from natural images using convolution neural networks," in *IEEE Workshop on Information Forensics and Security (WIFS)*, 2017.

[116] M. Zollhöfer, J. Thies, P. Garrido, D. Bradley, T. Beeler, P. Pérez, M. Stamminger, M. Nießner, and C. Theobalt, "State of the art on monocular 3d face reconstruction, tracking, and applications," *Computer Graphics Forum*, vol. 37, pp. 523–550, 2018.

[117] J. Thies, M. Zollhofer, M. Stamminger, C. Theobalt, and M. Nießner, "Face2face: Real-time face capture and reenactment of rgb videos," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2387–2395, 2016.

[118] J. Thies, M. Zollhöfer, and M. Nießner, "Deferred neural rendering: Image synthesis using neural textures," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–12, 2019.

[119] "Deepfakes github." `https://github.com/deepfakes/faceswap`.

[120] S. Agarwal, H. Farid, Y. Gu, M. He, K. Nagano, and H. Li, "Protecting world leaders against deep fakes," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019.

[121] S. Milani, M. Fontani, P. Bestagini, M. Barni, A. Piva, M. Tagliasacchi, and S. Tubaro, "An overview on video forensics," *APSIPA Transactions on Signal and Information Processing*, vol. 1, p. e2, 2012.

[122] P. Bestagini, S. Milani, M. Tagliasacchi, and S. Tubaro, "Codec and gop identification in double compressed videos," *IEEE Transactions on Image Processing (TIP)*, vol. 25, pp. 2298–2310, 2016.

[123] D. Vázquez-Padín, M. Fontani, D. Shullani, F. Pérez-González, A. Piva, and M. Barni, "Video integrity verification and gop size estimation via generalized variation of prediction footprint," *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 15, pp. 1815–1830, 2020.

[124] P. Bestagini, S. Battaglia, S. Milani, M. Tagliasacchi, and S. Tubaro, "Detection of temporal interpolation in video sequences," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[125] L. D'Amiano, D. Cozzolino, G. Poggi, and L. Verdoliva, "A patchmatch-based dense-field algorithm for video copy–move detection and localization," *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, vol. 29, pp. 669–682, 2019.

[126] M. Stamm, W. Lin, and K. Liu, “Temporal forensics and anti-forensics for motion compensated video,” *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 7, pp. 1315 –1329, 2012.

[127] A. Gironi, M. Fontani, T. Bianchi, A. Piva, and M. Barni, “A video forensic technique for detecting frame deletion and insertion,” in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6226–6230, 2014.

[128] R. A. Raimi, “The first digit problem,” *The American Mathematical Monthly*, vol. 83, no. 7, pp. 521–538, 1976.

[129] J. Wang, B. Cha, S. Cho, and C. . J. Kuo, “Understanding benford’s law and its vulnerability in image forensics,” in *2009 IEEE International Conference on Multimedia and Expo*, pp. 1568–1571, June 2009.

[130] A. Diekmann, “Not the first digit! using Benford’s law to detect fraudulent scientific data,” *Journal of Applied Statistics*, vol. 34, pp. 321–329, 04 2007.

[131] K.-H. Todter, “Benford’s law as an indicator of fraud in economics,” *German Economic Review*, vol. 10, pp. 339–351, 08 2009.

[132] S. Smoot and L. Rowe, “Study of DCT coefficient distributions,” in *SPIE Symposium on Electronic Imaging (EI)*, 1996.

[133] F. Pérez-González, T. T. Quach, C. Abdallah, G. L. Heileman, and S. J. Miller, “Application of benford’s law to images,” in *Benford’s Law: Theory and Applications* (S. Miller, ed.), ch. 19, Princeton University Press, 2015.

[134] C. Pasquini, G. Boato, and F. Pérez-González, “Statistical detection of JPEG traces in digital images in uncompressed formats,” *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 12, pp. 2890–2905, Dec 2017.

[135] S. Milani, M. Fontana, P. Bestagini, and S. Tubaro, “Phylogenetic analysis of near-duplicate images using processing age metrics,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016.

[136] S. S. Moin and S. Islam, “Benford’s law for detecting contrast enhancement,” in *International Conference on Image Information Processing (ICIIP)*, 2017.

[137] F. Matern, C. Riess, and M. Stamminger, “Exploiting visual artifacts to expose deepfakes and face manipulations,” in *IEEE Winter Applications of Computer Vision Workshops (WACVW)*, 2019.

[138] E. del Acebo and M. Sbert, “Benford’s law for natural and synthetic images.,” in *Eurographics Workshop on Computational Aesthetics in Graphics, Visualization and Imaging*, 2005.

[139] A. Makrushin, C. Kraetzer, T. Neubert, and J. Dittmann, “Generalized Benford’s law for blind detection of morphed face images,” in *ACM Workshop on Information Hiding and Multimedia Security (IH&MMSec)*, 2018.

[140] F. Marra, C. Saltori, G. Boato, and L. Verdoliva, “Incremental learning for the detection and classification of gan-generated images,” in *2019 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 1–6, 2019.

[141] H. Li, B. Li, S. Tan, and J. Huang, “Detection of deep network generated images using disparities in color components,” *arXiv preprint arXiv:1808.07276*, 2018.

[142] D. Güera and E. J. Delp, “Deepfake Video Detection Using Recurrent Neural Networks,” *IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*, 2019.

[143] E. Zakharov, A. Shysheya, E. Burkov, and V. Lempitsky, “Few-shot adversarial learning of realistic neural talking head models,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9458–9467, 2019.

[144] Y. Nirkin, Y. Keller, and T. Hassner, “Fsgan: Subject agnostic face swapping and reenactment,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7183–7192, 2019.

[145] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner, “FaceForensics++: Learning to detect manipulated facial images,” in *International Conference on Computer Vision (ICCV)*, 2019.

[146] “Deepfake Detection Challenge (DFDC).” https://deepfakedetectionchallenge.ai/, 2019.

[147] E. J. Delp, R. L. Kashyap, and O. R. Mitcheli, “Image data compression using autoregressive time series models,” *Pattern Recognition*, vol. 11, pp. 313–323, 1979.

[148] D. Minnen, J. Ballé, and G. D. Toderici, “Joint autoregressive and hierarchical priors for learned image compression,” in *Advances in Neural Information Processing Systems 31* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), pp. 10771–10780, Curran Associates, Inc., 2018.

[149] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," in *Proceedings of Machine Learning Research* (M. F. Balcan and K. Q. Weinberger, eds.), vol. 48, (New York, New York, USA), pp. 1747–1756, PMLR, 20–22 Jun 2016.

[150] A. van den Oord, N. Kalchbrenner, L. Espeholt, K. Kavukcuoglu, O. Vinyals, and A. Graves, "Conditional image generation with pixelcnn decoders," in *Advances in Neural Information Processing Systems 29* (D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, eds.), pp. 4790–4798, Curran Associates, Inc., 2016.

[151] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[152] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," in *International Conference on Learning Representations*, 2018.

[153] D. Cozzolino, D. Gragnaniello, and L. Verdoliva, "Image forgery localization through the fusion of camera-based, feature-based and pixel-based techniques," *Proceedings of the IEEE International Conference on Image Processing*, pp. 5302–5306, October 2014. Paris, France.

[154] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[155] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[156] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," in *NeurIPS*, 2018.

[157] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[158] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, 2016.

[159] V. Bazarevsky, Y. Kartynnik, A. Vakunov, K. Raveendran, and M. Grundmann, "Blazeface: Sub-millisecond neural face detection on mobile gpus," *CoRR*, vol. abs/1907.05047, 2019.

[160] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NIPS)* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), pp. 5998–6008, Curran Associates, Inc., 2017.

[161] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, "Residual attention network for image classification," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[162] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition*, pp. 7132–7141, 2018.

[163] H. Dang, F. Liu, J. Stehouwer, X. Liu, and A. Jain, "On the detection of digital face manipulation," 2019.

[164] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu, "Learning fine-grained image similarity with deep ranking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1386–1393, 2014.

[165] A. Buslaev, A. Parinov, E. Khvedchenya, V. I. Iglovikov, and A. A. Kalinin, "Albumentations: fast and flexible image augmentations," *ArXiv e-prints*, 2018.

[166] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), pp. 8024–8035, Curran Associates, Inc., 2019.

[167] D. Kingma and J. Ba, "Adam: a method for stochastic optimization. arxiv: 14126980," 2014.

[168] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

[169] H. E. Martz, C. M. Logan, D. J. Schneberk, and P. J. Shull, "X-ray imaging: Fundamentals, industrial techniques and applications," 2016.

[170] Food and A. O. of the United Nations., *Risk-based food inspection manual*. Food and Agriculture Organization of the United Nations, Rome, Italy, 2008.

[171] S. N. Jha, *Nondestructive evaluation of food quality: theory and practice*. Springer, 2010.

[172] D. Mery, I. Lillo, H. Loebel, V. Riffo, A. Soto, A. Cipriano, and J. M. Aguilera, "Automated fish bone detection using x-ray imaging," *Journal of Food Engineering*, vol. 105, no. 3, pp. 485–492, 2011.

[173] R. A. Speir and M. A. Haidekker, "Onion postharvest quality assessment with x-ray computed tomography – a pilot study," *IEEE Instrumentation Measurement Magazine*, vol. 20, pp. 15–19, June 2017.

[174] I. Khosa and E. Pasero, "Pine nuts selection using x-ray images and logistic regression," in *2014 World Symposium on Computer Applications Research (WSCAR)*, pp. 1–5, Jan 2014.

[175] M. Graves, A. Smith, and B. Batchelor, "Approaches to foreign body detection in foods," *Trends in Food Science and Technology*, vol. 9, no. 1, pp. 21–27, 1998.

[176] R. Haff and N. Toyofuku, "X-ray detection of defects and contaminants in the food industry," *Sensing and Instrumentation for Food Quality and Safety*, vol. 2, 12 2008.

[177] I. Khosa and E. Pasero, "Defect detection in food ingredients using multilayer perceptron neural network," in *2014 World Symposium on Computer Applications Research (WSCAR)*, pp. 1–5, Jan 2014.

[178] I. Khosa and E. Pasero, "Feature extraction in x-ray images for hazelnuts classification," in *2014 International Joint Conference on Neural Networks (IJCNN)*, pp. 2354–2360, July 2014.

[179] Q. Lu, F. Wang, Y. Li, and J. Cai, "Real-time nondestructive inspection of chestnuts using x-ray imaging and dynamic threshold," in *2010 World Automation Congress*, pp. 365–368, Sep. 2010.

[180] A. D. M. Naught and A. Wilkinson, *IUPAC Compendium of Chemical Terminology – The Gold Book*. Blackwell Scientific Publications, Oxford, 1997.

[181] M. Wocher, K. Berger, M. Danner, W. Mauser, and T. Hank, "Hyperspectral retrieval of canopy water content through inversion of the beer-lambert law," in *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, pp. 3805–3808, July 2018.

[182] S. Kelkar, C. J. Boushey, and M. Okos, "A method to determine the density of foods using x-ray imaging," *Journal of food engineering*, vol. 159, pp. 36–41, 2015.

[183] G. Wypych, "Photophysics," in *Handbook of Material Weathering*, pp. 1–26, Elsevier, 2018.

[184] D. J. Griffiths, *Introduction to Quantum Mechanics*. Prentice Hall, 1994.

[185] R. C. Murty, "Effective atomic numbers of heterogeneous materials," *Nature*, vol. 207, 1965.

[186] G. N. Lewis, "The conservation of photons," *Nature*, vol. 118, pp. 874–875.

[187] L. Mandel, "Fluctuations of photon beams: The distribution of the photo-electrons," *Proceedings of the Physical Society*, vol. 74, pp. 233–243, sep 1959.

[188] N. Wiener, *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. The MIT Press, 1964.

[189] A. Hiller and R. Chin, "Iterative wiener filters for image restoration," in *International Conference on Acoustics, Speech, and Signal Processing*, pp. 1901–1904 vol.4, 1990.

[190] A. M. A. Mahmood, "Optimized and iterative wiener filter for image restoration," in *2009 6th International Symposium on Mechatronics and its Applications*, pp. 1–6, 2009.

[191] V. Katkovnik, A. Foi, K. Egiazarian, and J. Astola, "From local kernel to nonlocal multiple-model image denoising," *International Journal of Computer Vision*, vol. 86, pp. 1–32, jul 2009.

[192] C. Cruz, R. Mehta, V. Katkovnik, and K. O. Egiazarian, "Single image super-resolution based on wiener filter in similarity domain," *IEEE Transactions on Image Processing*, vol. 27, pp. 1376–1389, mar 2018.

[193] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[194] L. Gondara, "Medical image denoising using convolutional denoising autoencoders," in *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pp. 241–246, Dec 2016.

[195] H.-T. Chiang, Y.-Y. Hsieh, S.-W. Fu, K.-H. Hung, Y. Tsao, and S.-Y. Chien, "Noise reduction in ecg signals using fully convolutional denoising autoencoders," *IEEE Access*, vol. 7, pp. 60806–60813, 2019.

[196] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[197] Z. Jing, B. Wu, P. Li, R. Yang, J. Li, and Z. Wang, "A novel variational autoencoder based radar signal reconstruction algorithm using polluted data," in *IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium*, pp. 2715–2718, 2020.

[198] H. Wu, X. Ma, and S. Liu, “Designing multi-task convolutional variational autoencoder for radio tomographic imaging,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, pp. 1–1, 2021.

[199] P. Ghosh, M. S. M. Sajjadi, A. Vergari, M. Black, and B. Scholkopf, “From variational to deterministic autoencoders,” in *International Conference on Learning Representations*, 2020.

[200] C. Tian, Y. xu, L. Fei, and K. Yan, “Deep learning for image denoising: A survey,” *preprint arXiv:1810.05052,2018*, 10 2018.

[201] C. Cruz, A. Foi, V. Katkovnik, and K. Egiazarian, “Nonlocality-reinforced convolutional neural networks for image denoising,” *IEEE Signal Processing Letters*, vol. 25, pp. 1216–1220, aug 2018.

[202] F. Kokkinos and S. Lefkimmiatis, “Deep image demosaicking using a cascade of convolutional residual denoising networks,” in *Computer Vision – ECCV 2018*, pp. 317–333, Springer International Publishing, 2018.

[203] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, “Extracting and composing robust features with denoising autoencoders,” in *Proceedings of the 25th international conference on Machine learning - ICML '08*, ACM Press, 2008.

[204] J. Xie, L. Xu, and E. Chen, “Image denoising and inpainting with deep neural networks,” in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 341–349, Curran Associates, Inc., 2012.

[205] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, “Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion,” *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, Dec. 2010.

[206] J. Xu and B. Nguyen, “CMOS image sensor based x-ray detector noise characterization and its fixed pattern noise correction method,” in *Medical Imaging 2011: Physics of Medical Imaging* (N. J. Pelc, E. Samei, and R. M. Nishikawa, eds.), SPIE, mar 2011.

[207] F. Chollet *et al.*, “Keras.” `https://keras.io`, 2015.

[208] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, “A survey on deep transfer learning,” in *ICANN*, 2018.

[209] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[210] B. Xu, N. Wang, T. Chen, and M. Li, “Empirical evaluation of rectified activations in convolutional network,” *arXiv preprint arXiv:1505.00853*, 2015.

[211] A. A. Hendriksen, D. M. Pelt, and K. Joost Batenburg, “Noise2Inverse: Self-supervised deep convolutional denoising for tomography,” *arXiv e-prints*, Jan. 2020.

[212] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila, “Noise2Noise: Learning image restoration without clean data,” in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, (Stockholmsmässan, Stockholm Sweden), pp. 2965–2974, PMLR, 10–15 Jul 2018.

[213] M. Re, “Denoising and classification of hyperspectral X-ray images for food quality assessment,” *POLITesi - Politecnico di Milano*, 2018.

[214] D. Clevert, T. Unterthiner, and S. Hochreiter, “Fast and accurate deep network learning by exponential linear units (elus),” in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2016.

[215] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015.