Towards model-based flexible and adaptive image forensics

Ludovic Darmet

UNIVERSITÉ GRENOBLE ALPES

THÈSE

pour obtenir le grade de
DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE ALPES

Spécialité : **SIGNAL, IMAGE, PAROLE, TÉLÉCOMS**

Arrêté ministériel : 25 mai 2016

Présentée par
Ludovic DARMET

Thèse dirigée par **François CAYRE** et co-encadrée par **Kai WANG**

préparée au sein du
Grenoble Images Paroles Signal Automatique (GIPSA-lab)
dans **l'école doctorale EEATS (ED EEATS)**

Towards model-based flexible and adaptive image forensics

Vers une approche basée modèle-image flexible et adaptative en criminalistique des images

Thèse soutenue publiquement le 1er **décembre 2020**,
devant le jury composé de :

Michèle ROMBAUT
GIPSA-lab, Université Grenoble Alpes, Grenoble, Présidente
Luisa VERDOLIVA
Department of Industrial Engineering, University of Naples Federico II, Naples, Rapporteuse
Jean-Michel MOREL
Centre Borelli, ENS Cachan, Cachan, Rapporteur
Patrick BAS
CRIStAL, CNRS, Lille, Examinateur
François CAYRE
GIPSA-lab, Grenoble INP, Grenoble, Directeur de thèse
Kai WANG
GIPSA-lab, CNRS, Grenoble, Co-encadrant, Invité

Abstract

Images are nowadays a standard and mature medium of communication. They appear in our day to day life and therefore they are subject to concerns about security. In this work, we study different methods to assess the integrity of images. Because of a context of high volume and versatility of tampering techniques and image sources, our work is driven by the necessity to develop flexible methods to adapt the diversity of images.

We first focus on manipulations detection through statistical modeling of the images. Manipulations are elementary operations such as blurring, noise addition, or compression. In this context, we are more precisely interested in the effects of pre-processing. Because of storage limitation or other reasons, images can be resized or compressed just after their capture. Addition of a manipulation would then be applied on an already pre-processed image. We show that a pre-resizing of test data induces a drop of performance for detectors trained on full-sized images. Based on these observations, we introduce two methods to counterbalance this performance loss for a pipeline of classification based on Gaussian Mixture Models. This pipeline models the local statistics, on patches, of natural images. It allows us to propose adaptation of the models driven by the changes in local statistics. Our first method of adaptation is fully unsupervised while the second one, only requiring a few labels, is weakly supervised. Thus, our methods are flexible to adapt versatility of source of images.

Then we move to falsification detection and more precisely to copy-move identification. Copy-move is one of the most common image tampering technique. A source area is copied into a target area within the same image. The vast majority of existing detectors identify indifferently the two zones (source and target). In an operational scenario, only the target area represents a tampering area and is thus an area of interest. Accordingly, we propose a method to disentangle the two zones. Our method takes advantage of local modeling of statistics in natural images with Gaussian Mixture Model. The procedure is specific for each image to avoid the necessity of using a large training dataset and to increase flexibility.

Results for all the techniques described above are illustrated on public benchmarks and compared to state of the art methods. We show that the classical pipeline for manipulations detection with Gaussian Mixture Model and adaptation procedure can surpass results of fine-tuned and recent deep-learning methods. Our method for source/target disentangling in copy-move also matches or even surpasses performances of the latest deep-learning methods. We explain the good results of these classical methods against deep-learning by their additional flexibility and adaptation abilities.

Finally, this thesis has occurred in the special context of a contest jointly organized by the French National Research Agency and the General Directorate of Armament. We describe in the Appendix the different stages of the contest and the methods we have developed, as well as the lessons we have learned from this experience to move the image forensics domain into the wild.

Résumé

Les images numériques sont devenues un moyen de communication standard et universel. Elles prennent place dans notre vie de tous les jours, ce qui entraîne directement des inquiétudes quant à leur intégrité. Nos travaux de recherche étudient différentes méthodes pour examiner l'authenticité d'une image numérique. Nous nous plaçons dans un contexte réaliste où les images sont en grandes quantités et avec une large diversité de manipulations et falsifications ainsi que de sources. Cela nous a poussé à développer des méthodes flexibles et adaptative face à cette diversité.

Nous nous sommes en premier lieu intéressés à la détection de manipulations à l'aide de la modélisation statistiques des images. Les manipulations sont des opérations élémentaires telles qu'un flou, l'ajout de bruit ou une compression. Dans ce cadre, nous nous sommes plus particulièrement focalisés sur les effets d'un pré-traitement. A cause de limitations de stockage et autres, une image peut être re-dimensionnée ou re-compressée juste après sa capture. L'ajout d'une manipulation se fait donc ensuite sur une image déjà pré-traitée. Nous montrons qu'un pré-redimensionnement pour les images de test induit une chute de performance pour des détecteurs entraînés avec des images en pleine taille. Partant de ce constat, nous introduisons deux nouvelles méthodes pour mitiger cette chute de performance pour des détecteurs basés sur l'utilisation de mixtures de gaussiennes. Ces détecteurs modélisent les statistiques locales, sur des tuiles (patches), d'images naturelles. Cela nous permet de proposer une adaptation de modèle guidée par les changements dans les statistiques locales de l'image. Notre première méthode est une adaptation entièrement non-supervisée, alors que la seconde requière l'accès à quelques labels, faiblement supervisé, pour les images pré-resizées.

Ensuite, nous nous sommes tournés vers la détection de falsifications et plus spécifiquement l'identification de copier-coller. Le copier-coller est l'une des falsification les plus populaires. Une zone source est copiée vers une zone cible de la même image. La grande majorité des détecteurs existants identifient indifféremment les deux zones (source et cible). Dans un scénario opérationnel, seulement la zone cible est intéressante car uniquement elle représente une zone de falsification. Ainsi, nous proposons une méthode pour discerner les deux zones. Notre méthode utilise également la modélisation locale des statistiques de l'image à l'aide de mixtures de gaussiennes. La procédure est spécifique à chaque image et ainsi évite la nécessité d'avoir recours à de larges bases d'entraînement et permet une plus grande flexibilité.

Des résultats expérimentaux pour toutes les méthodes précédemment décrites sont présentés sur des benchmarks classiques de la littérature et comparés aux méthodes de l'état de l'art. Nous montrons que le détecteur classique de détection de manipulations basé sur les

mixtures de gaussiennes, associé à nos nouvelles méthodes d'adaptation de modèle peut surpasser les résultats de récentes méthodes deep-learning. Notre méthode de discernement entre source/cible pour copier-coller égale ou même surpasse les performances des dernières méthodes d'apprentissage profond. Nous expliquons ces bons résultats des méthodes classiques face aux méthodes d'apprentissage profond par la flexibilité et l'adaptabilité supplémentaire dont elles font preuve.

Pour finir, cette thèse s'est déroulée dans le contexte très spécial d'un concours organisé conjointement par l'Agence National de la Recherche et la Direction Général de l'Armement. Nous décrivons dans un appendice, les différents tours de ce concours et les méthodes que nous avons développé. Nous dressons également un bilan des enseignements de cette expérience qui avait pour but de passer de benchmarks publics à une détection de falsifications d'images très réalistes.

Remerciements

Tout d'abord et bien évidemment, merci à mes encadrants François et Kai sans qui rien de tout ça n'aurait pu arriver. Vous m'avez laissé beaucoup de liberté tout en étant là pour m'aider quand j'en avais besoin. Je vous remercie beaucoup pour cela. Plus spécifiquement, merci à Kai pour ta patience devant mes nombreuses erreurs stupides d'implémentation, pour toutes ces discussions dans ton bureau et pour les relectures nombreuses et attentives. Merci à toi François de m'avoir aidé à trouver la direction pendant cette thèse avec notamment les discussions sur ce banc devant le GIPSA. Merci aussi pour tes relectures toujours pertinentes. Merci à Patrick Bas, pour avoir d'une part permis les conditions matérielles de cette thèse et pour le challenge DEFALS qui aura été une bonne inspiration.

Ensuite je voudrais remercier les membres de mon jury. Merci à Jean-Michel Morel et Luisa Verdoliva pour la relecture attentive de mon manuscrit et les remarques détaillées. Merci encore à Patrick Bas pour ton rôle d'examinateur, ces discussions et questions. Enfin merci à Michèle Rombaut d'avoir accepté de jouer ce rôle de présidente du jury et en anglais. Je suis fière d'avoir soutenu devant un tel jury !

Merci à tous les copains pour ces 3 belles années.
Ceux du GIPSA : Julien (malgré tes goûts musicaux douteux #Skrillex), Bruce (le travail conjoint pour vider le distrib de la cafèt), Ivan (aussi bien pour DEFALS que pour le reste), Dawood (thanks for my English skills), Maria, Pedro (sans qui il n'y aurait pas le chapitre principal...), Tien, Jitu, Isidora, Hélène, Omar, Cosme, Malik et tout la bande à Couillet, Jeanne et tous les autres.
Les Gre'lou pour tout ces chouettes moments sur le caillou, la neige et autour de prises en résine : Pierre, Guillaume, Hugo, Adri, Tom, Emma, Manu, Benj, Thibault et tous ceux qui ont été à l'autre bout de la corde un jour.
Ceux du lycée pour me rappeler de pas trop prendre tout ça au sérieux : Clément, Delphine, Romain, Mick et tous les autres.
Ceux de Centrale pour les week-ends intellectuels : les Casports, Boubou et un s/o spécial à

Max et Tits pour l'inspiration à se lancer dans la thèse !

Merci aussi à ma famille pour l'enthousiasme et le soutien, s/o particulièrement à mon frère Yvan pour les figures d'images falsifiées !

Enfin merci beaucoup à celle qui a le plus subi les hauts et les bas de la thèse avec toujours beaucoup de patience et de réconfort. Tu as aussi accepté tous ces allers-retours le week-end. Sans ton soutien Flore-Anne je n'aurais jamais pu faire tout ça.

Contents

1 | Introduction

Images have always been a privileged medium for communication. Children are able to unscramble images long before they are able to read a text. Before invention of writing, usually dated 5000 years ago, transmission was mainly oral and also through pictures. Prehistory paintings in caves are known worldwide. However, one popular motto states that *"a picture is worth a thousand words*". It could be understood from this motto that sometimes oral is not efficient enough to transmit information and images are required. For instance, on the 5^{th} of February 2003, U.S. Secretary of State Colin Powell during his hearing in front of the UN Security Council has disclosed some satellite images that were supposed to be evidence regarding the development of chemical weapons in Iraq, which would justify a military intervention. M. Powell had also exhibited phone taping and other materials, though the images have presumably played an important role. This shows the important role of images in decision-making processes. Generally speaking, images generate *affects* that steer decisions keenly.

Following the question of the power of images and the affects they can generate, appears immediately the question of the trust we can put in them. Before introducing image tampering, one should note that an image hardly comes out of the blue for the sake of itself. It is usually presented by someone in a context. The context itself can be misleading. One example can be found on Figure 1.1. The Republicans, M. Trump political party, and his supporters had shared mostly the image on the left (a), acknowledging that a big crowd was attending the 2017 presidential inauguration. Yet, his political opponents preferred to disseminate the picture on the right (b), so the unflattering comparison with M. Obama's inauguration in 2012 would become obvious, asserting that the crowd was much bigger back then. None of the pictures in Figure 1.1 have been altered and they both portray the same genuine scene from 2017. However, the affects induced are very different. It should be the role of the journalists to present the context precisely and provide accurate pictures to allow more rational decision, with less affects.

(a) (b)

Figure 1.1 – Picture of crowds in front of the Capitol during public inauguration of M. Donald Trump to US president in 2017.

We have reminded at the beginning of this chapter that the war in Iraq in 2003 had been to a certain extent justified by some satellite images of alleged Iraqi plants of chemical weapons. It was discovered later that these images were fake. Some of the evidence that had justified a military intervention in Iraq were made up by US administration. This raises particularly the question of trust in images. Context and angle selection for an image could mislead the interpretation like we discussed just before, yet it is even far worse when the image gets tampered.

In our societies, it is necessary to build reliable tools to assess the integrity of images. Concerns about image alterations are not exactly new, one of the most known tampered image being the one with Stalin and the removed colonel Yezhov (see Figure 1.2). Yet, these concerns have only been made even more worrisome with the recent advent of digital images. The wide availability of smartphones and digital cameras, combined with social platforms, has made sharing images easier than ever. Moreover, image editing software has become easy to use, even for the layman, and it is now available on smartphones. One can therefore capture, modify and share an image quickly with the same handheld device. This situation both accounts for a tremendous revolution in communication and makes it harder than ever to trust images, as their integrity may be altered easily, almost on the fly. Hence, forensics tools are being developed to help assess the source and integrity of digital images.

These forensics tools can be either automatic or require the help of a specialist. The need for specialists has become quite common. Most major newspapers have a "*fact-checking*" team that is able to investigate the truth of a piece of information. For instance, they are called

(a) (b)

Figure 1.2 – Official picture from USSR regime of Stalin and close advisers (1940). Colonel Yezhov was removed on picture (b) after Stalin held him in disgrace.

Les Décodeurs for the French journal *Le Monde*. Their work is closely related to the work of journalists discussed earlier, as their investigations are usually led by the context. Indeed, they are part of the newspaper. However, such an approach can present some problems or limitations. Namely, we should put trust in these teams and their skills. Yet, they are also humans with bias and that could be affected, even though they surely try their best to be as neutral as possible. As their methods are mostly non automatic, they can process only a limited number of pictures. Therefore, the choice of which picture to investigate is already a bias. To reduce this bias, one should be able to process the more images as possible. This calls for automatic tools that should be able to process a large number of images, without further introducing any human bias in it.

1.1 Overview

This thesis presents contributions in the field of **image forensics**. It concerns the security of digital images and acts as a *passive* image authentication approach. It contrasts with the close field of *watermarking* which implements an *active* image authentication [1]. A hidden, fragile *watermark* would be added into any picture directly by the device and testing for the watermark integrity would in turn help conclude whether the image had undergone a manipulation or a falsification. The inspection of images is then straightforward, although standardized implementation of fragile watermarking in each and every camera has never

1. Meaning that a watermark has to be actively embedded in the original image prior to dissemination.

been contemplated by the industry.

Images are composed of pixels and metadata. Pixels are arranged in a grid of integers and the metadata will usually describe which camera took the picture and under which optical conditions. The study of metadata may partially inform about the integrity of images. Although fingerprints of modifications in metadata would differ considerably regarding the considered dataset. These metadata are also easily editable and therefore should be considered unreliable. For these two reasons, metadata have been less investigated and most existing methods focus on pixels.

Common image forensics problems include camera identification, image manipulation detection, identification of computer graphics images, splicing detection, copy-move detection, identification of fake faces, *etc.* Camera identification is quite specific and the remaining problems basically fall in two categories: manipulation and falsification detection. Manipulations are usually elementary operations, while falsifications target more elaborated tampering. Therefore, it is usually easier to detect manipulations. It has been the prime interest of the field at its beginning approximately 10 years ago. Then the problems can be:

— Binary classification of the whole image (manipulated or not, falsified or not),
— Identification of the specific manipulation or falsification (*e.g.*, identifying Gaussian blurring or copy-move),
— Localization of the forgeries.

These three problems are clearly related and present different levels of complexity. The field now concentrates mainly on localization. Localization is also necessary to be able to evaluate the impact of the falsification on the semantics of the image.

1.2 DEFALS contest

This thesis has been supported by an ANR-DGA grant, in the context of the DEFALS contest (DEtection de FaLSifications dans des images). The aims of this contest are:

— To initiate and stimulate research in image forensics (passive detection of falsification in realistic forged images);
— To foster collaboration between academic research in computer vision, users and industrial partners.

Four French teams were selected to take part in the contest. We are part of the REVEAL team, led by Patrick BAS from CRIStAL (Lille). Other team members are Ivan Castillo-Camacho, a PhD student in GIPSA-lab, Kai Wang, a researcher in GIPSA-lab, François Cayre, an associate professor in GIPSA-lab, and Gaëtan Le Guelvouit from B<>Com (Rennes). B-

Com is the industrial partner of the project, responsible for the integration of the techniques developed by other members. Other teams are SIGNATURE from ENS-Cachan (Cachan); OEIL from LIRMM (Montpellier); and DEFACTO from UTT (Troyes).

It is a contest between the four teams in several rounds. The first round took place in February 2018, as a warm-up, preliminary round. The aim was the binary classification between falsified and original images. The first official round in March 2019 was also about binary classification of falsified images, on another set of images. Finally, the last round took place in March and April 2020 and the aim was localization of image falsifications. What makes this contest outstanding is that images are particularly realistic. Indeed, image manipulations have been carefully performed by professionals on high resolution (3000×4000 pixels) images. Falsifications come in several types (copy-move, splicing, inpainting, text replacement). Most of the other, public databases (see Section 2.2) are in lower resolution (usually less than 750×1000 pixels) and falsifications are performed in an automatic way which makes it easier to detect and less realistic. Most of the existing methods from the literature failed on the DEFALS datasets.

This contest has highly influenced the conduct of the thesis. Firstly, it has helped to point out the importance of flexible methods as for each new round or dataset, detectors should be adapted to perform well. Another important note is that the contest has greatly constrained the planning of the thesis. The two official rounds have required about two months of preparation with the training data, and one month for the contest phase. The preliminary round was a warm-up and took us about a month and a half. In total it has occupied our time and our computing resources about 20% of the thesis. It was a bit disappointing as the results can not directly be harnessed to develop new methods. To perform well in a contest, methods are very specific to the dataset. For instance, we worked a lot on metadata which is not generalizable. A lot of time was also devoted to engineering and re-implementation of existing tools. A famous example is the Netflix Price, a recommendation competition hosted by Netflix with 1 million dollars prize. As Netflix had explained in their technical blog [Net], they never used the winning algorithm to the contest because *"additional accuracy gains that we measured did not seem to justify the engineering effort*". However, it was a pretty unique opportunity in this thesis to work on such high-quality, real-world datasets.

We won the preliminary and the first rounds. For the second and last round, we were still the best in binary classification, but the third in localization.

1.3 Objectives of the thesis

Ever since the last ten years, very efficient methods have been developed on public benchmarks. Some commercial software tools have been released, which nonetheless require intervention of a highly-skilled user. Yet, no commercial software exists today that performs automatically for images in the wild. The diversity of image parameters (source, size, quality, compression, pre-processing, *etc.*) is considerable and each of them influences the performances of most existing detectors. On top of that, it also exists a tremendous variety of manipulations and falsifications which are performed using heterogeneous software. That could explain the difficulty to extend efficient methods on public benchmarks to the diversity of images in the wild. Rather than looking for a method that could generalize perfectly over all this variety, our strategy is to develop **adaptation strategies** that allow the use of already developed and powerful models with different testing data or new objective. We feel that flexibility of models is a crucial point to be able to move image forensics from the laboratory to the wild.

In short, the main objective of this thesis is to develop adaptation methods on top of already existing detectors. Adaptation should be data-driven and specific for a testing dataset. This objective has been partially motivated by the poor performance we witnessed when using state of the art methods on the highly realistic DEFALS datasets. To the best of our knowledge, this approach is new and no other existing adaptation approaches have been considered yet in image forensics.

1.4 Organization of the manuscript

In Chapter 2, we propose an overview of the field of Image Forensics and the state of the art approaches for the two different problems of manipulation and falsification detection. We also present briefly some basics of *Machine Learning* as we use methods and concepts from this framework in our methods.

Then in Chapter 4, we introduce two adaptation strategies for image manipulation detection. They are applied on a detector based on Gaussian Mixture Model (GMM) developed by Wei Fan *et al* [FWC15]. This detector aims to detect local statistical deviations in the image, on small patches, that would indicate a manipulation. Basically, the GMM is composed of two sets of parameters: weights and covariance matrices. The first method focuses on weights adaptation with the help of a few labels on the target dataset (weakly supervised approach). We have communicated about this work during the 18^{th} International Workshop

on Digital Forensics and Watermarking (Chengdu, China) [DWC19b]. The second method is named *GRAFT* and deals with the covariance matrices of the GMMs, without the help of labels on the target dataset (unsupervised approach). This method has been published in the IEEE Access [DWC20] journal and also as a communication at the GRETSI [DWC19a] French national conference.
We have begun with a focus on image manipulation, as we consider that more sophisticated falsifications are composed of elementary manipulations. For instance, a copy-move is often associated with a smoothing of the tampered area border through Gaussian blurring. Therefore, being able to detect elementary manipulations should allow us to detect more elaborated falsifications. On another note, manipulations are often easier to detect than falsifications and as we considered a new approach (adaptation), we chose to focus on manipulation detection rather than falsification. We also consider that falsifications and manipulations are related, as both introduce perturbations in local statistics of the image. Hence, this work on adaptation for manipulation detector is expected to be useful also for falsification detectors.

In Chapter 5, we switch our focus to falsification and more specifically on copy-move detection. In a copy-move, a part of an image, the source, is copied somewhere else within the same image, as the so-called target area. It is probably one of the most popular falsifications along with splicing, where the copied part comes from another image. There already exist powerful copy-move detectors. However, most of these methods do not identify which area is the source and which is the target in the copy-move. Thus, we have developed a method on top of these detectors to adapt them to distinguish between the two areas. This second stage of detection could also help detect false positives of the first-stage detector. Our method is intended to work with any first-stage copy-move detector. There is a direct link with our previous work on manipulations detection, as it also relies on deviations of the local statistics for detection (this time on the boundaries and interiors of the source and target areas). The method is flexible by design as it is specific for each image. Thus a large database of copy-move images with ground-truth masks is not required. This is to a certain extent similar to the adaptation of manipulation detector presented in Chapter 4. The underlying idea is to make the forensics detector adaptive to the testing data, without performing heavy training on a large amount of diverse data. We have submitted a paper describing our proposal to a special issue of the Elsevier journal Applied Soft Computing called “Applying Machine Learning for Combating Fake News and Internet/Media Content Manipulation”.

Finally, we draw conclusions in Chapter 6 regarding the undertaken investigations of this thesis. We also discuss future possible lines of research for Image Forensics and especially regarding adaptation of models and features.

In Appendix A, we present the DEFALS challenge and the different rounds of the contest.

This chapter is mainly about practical detection in a highly realistic scenario. More precisely, we discuss the importance of meta-data in this kind of scenario and the fusion of decisions from multiple detectors. This work has highlighted the value of ranking the certainty of the predictions. In a very realistic scenario, performances are downgraded by many false positives and negatives. With a ranking, it is possible to extract subsets with high performances.

2 | Image Forensics

Contents

2.1 Introduction to Image Forensics

As introduced in the previous Chapter 1, **image forensics** relates to passive digital image authentication. The overall aim is to assess integrity of an image originating from an unknown source. Several approaches have been used to this end:

1. Searching for clues in meta-data [MRF19]; [Fac12]; [KJF11]; [GKR13];
2. Investigating inconsistencies in lightening or reflection [Fan+12]; [OF12]; [KOF14]; [KOF13];
3. Searching for tampered fingerprints in pixel values.

Approach 1. is probably the least reliable as meta-data can be changed or erased very easily and without leaving any traces. Moreover, since meta-data clues are dataset-dependent, it is hard to devise a general method from them. Approach number 2. considers images through geometrical and physical points of view. It typically implies to *(i)* first build a physical model for lightening, reflection, or shading, and *(ii)* then detect lighting or reflection sources in the image to further apply the inferred model on it. Image tampering would likely expose inconsistencies with the model. Physical models can be complex in case of multiple lightnings or reflection sources through sub-optimal surfaces such as turbulent sea. Some images do not even have any reflections or shades. Because of these difficulties, it has been slightly less investigated than methods from the third approach. The last, third approach is the most studied. This work, in line with the following state of the art, focuses on these methods.

2.1.1 Digital image development process

Before presenting an overview of methods for manipulation or falsification detection, it seems necessary to remind how an image is generally captured, from the scene to the file stored on the camera memory card.

First, the light emanating from the physical scene passes through multiple lenses. These lenses focus light on the sensors of the camera. In addition to the lenses some filters are sometimes added to remove polarized light (for pictures with snow or water for instance) or other types of light for artistic purposes. Then, the light passes through a *color filter array* (CFA). It is a physical filter composed of a mosaic of color filters to spatially separate colors. Indeed, classical sensors are not able to separate color information as they are not specific to any particular wavelength. With the CFA, each pixel of the sensor capture intensity information regarding only a specific color. Pixels are elementary components of digital photo-sensors. They are arranged through a grid to compose a photo-sensor. It is the CFA which allows the

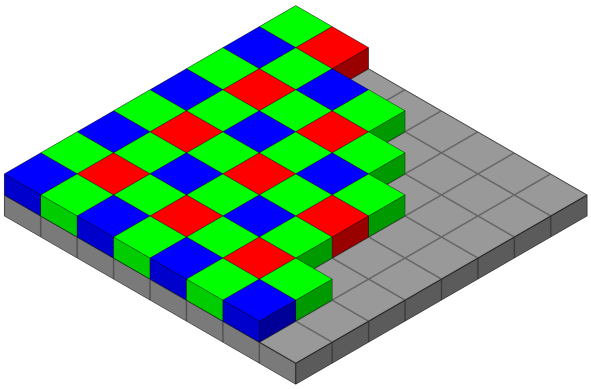


Figure 2.1 – The color mosaic for the Bayer filter (Source: Wikipedia — https://en.wikipedia.org/wiki/Bayer_filter). The incoming light is filtered by mosaic color filter before hitting the pixel array of the image sensor, so each pixel only captures one color.

camera to capture color images: without it, only grayscale images would be obtained. There exist several patterns for the CFA, depending on the camera manufacturer. The most common one is the Bayer filter, depicted on Figure 2.1.

An interpolation is then performed to recover the missing color information for every pixel. The interpolation algorithm is also specific to the manufacturer. After this interpolation step and depending on the camera, software routines perform basic operations to enhance image, such as white balance, contrast saturation, *etc*. For reflex camera it is however often possible to retrieve picture in *RAW* formats. An image in the RAW format has neither been interpolated nor enhanced. It is the direct, raw output from the sensor. With smartphones or cheaper cameras, the image is usually compressed to be stored. The most commonly used algorithm for natural image compression is the *JPEG* algorithm. JPEG algorithm has several hyper-parameters, including a quantization table that could be either standard or, again, specific to the manufacturer. All these steps between the scene and the image outputted by the camera are summarized on Figure 2.2.

IIn brief, capturing an image through a digital camera not only implies physical devices such as lenses and light sensors, but also software operations for color interpolation, basic processing and compression. The physical devices produce a noise pattern that is specific to a manufacturer and a camera model but also to the very camera at hand, as each sensor and set of lenses are unique because of means of production. The software also introduces specific fingerprints that would vary between cameras of different brands or types (camera of smartphones, compact cameras, reflex cameras, *etc*).
This variability is both an opportunity and a weakness. It could allow forensics investigators to identify pictures from a specific brand or even a specific camera thanks to the particular

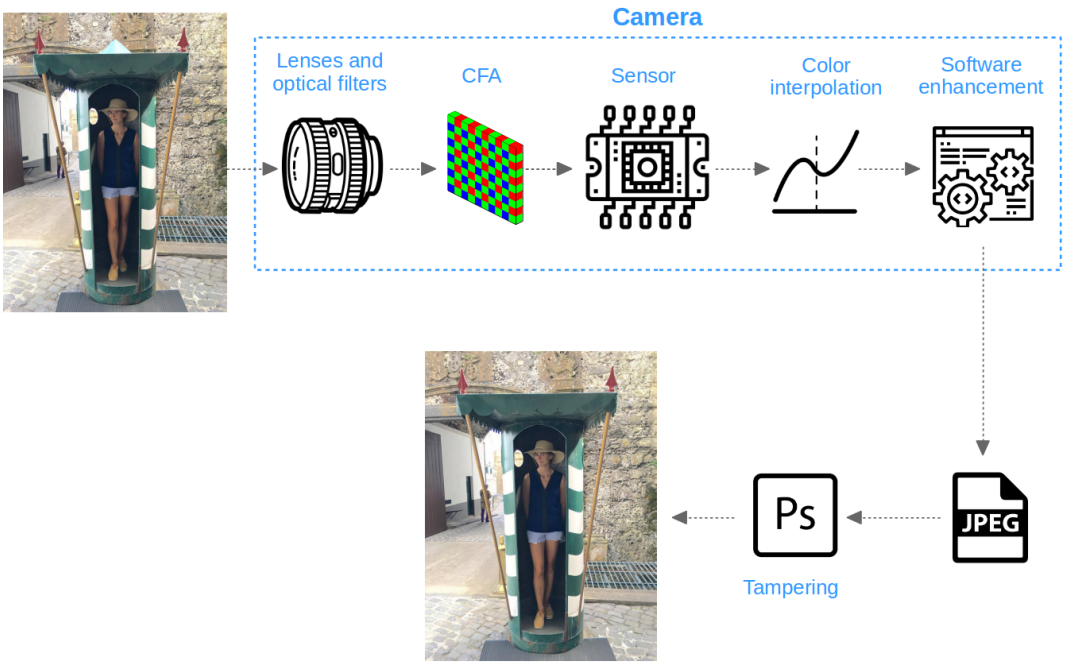


Figure 2.2 – An example of a typical development process of a digital image, with a tampering.Starting from the physical scene, image development occurs within the camera to finally produce a representation of the image in digital format. Tampering occurs afterwards. In this example, the tampering is the removal of the pyramid on the top of the cabin by using a Photoshop inpainting tool called *Clone Stamp Tool*. The figure is largely inspired by [Piv13].

fingerprints from physical elements (particularly the sensor) and the software. On the other hand, these fingerprints and noises could interfere with or even partially mask fingerprints of manipulation or falsification. They should be disentangled. It also creates a lot of diversity in images that brings complexity for modeling.

2.1.2 Manipulation detection

We would like to emphasize the distinction between image **manipulation** and **falsification**. We define manipulation as the application of an elementary operation, performed on the image globally. Typical examples are JPEG compression, noise addition, contrast enhancement, blurring, *etc*. Ordinarily, such operations do not change the semantics of the image. We define falsification as more elaborated tampering which is local. Typical examples are copy-move, inpainting and addition of external elements. Falsifications usually change the semantics of the image. Manipulations can be combined with falsification to cover tampering fingerprints.

Two main types of methods have been developed for detection of image manipulation (elementary operations):

1. Methods aimed at detecting a targeted and specific manipulation [Yua11]; [Kan+13]; [Fd03]; [LHQ10]; [Cao+14]; [Cao+11]; [Che+15];
2. More recently, universal detectors of image modifications (*i.e.* same model/features for every modifications) have been designed.

In order to tackle the second and more difficult problem of universal detection, three approaches have been followed:

1. Explicit modeling of images and spotting inconsistencies with respect to the model [FWC15];
2. Computing hand-crafted features (mainly borrowed from steganalysis such as SPAM [PBF10] and SRM [FK12]) as statistics of an implicit model and using a classifier on the feature space [Qiu+14]; [Li+18];
3. Using deep learning classifiers with constraints or specific processing for the first layer in order to extract residuals [BS16]; [Che+15].

This work focuses on developing universal detectors of image modifications through explicit modeling (Approach 1.). Other approaches (2. and 3.) are also considered for comparison purposes. It appears quite reasonable to us that a detector should rather be universal than targeted as the diversity of manipulations would eventually keep growing. Beside that, considering the huge progress made in the deep learning field in the recent years, we reckon that methods relying on hand-crafted features are less competitive. Deep-learning methods jointly and automatically optimize the features construction and the classifier training. Methods based on hand-crafted features are still the best ones for explainability but not in terms of pure performance. A similar claim about explainability and pure performance can be made for deep learning and explicit modeling of images. Quite logically, the trend now in the field of image manipulation detection is towards deep learning methods.

While pure performance of deep-learning is no longer necessary to prove, it usually relies on very large labeled databases. We also reckon that these methods are not really flexible. As explained earlier, our interest was more in terms of flexibility to adapt to the diversity of images in the wild than pure performance on a specific dataset. This has motivated us to focus on explicit modeling rather than deep-learning. We have considered explicit modeling over hand-crafted features because we think it would be easier to adapt explicit model than feature-based detectors as the link with statistics of data is really direct. It is usually challenging to find transformations of features to adapt to new statistics of data. In contrast, the classical approach relies on re-starting a manual labelling of the new data an then re-training the detector on features extracted from the data with the new statistics. Recent efforts have

been made to partially avoid or alleviate this costly re-labelling and re-training [PY10], also known as *Transfer Learning*.

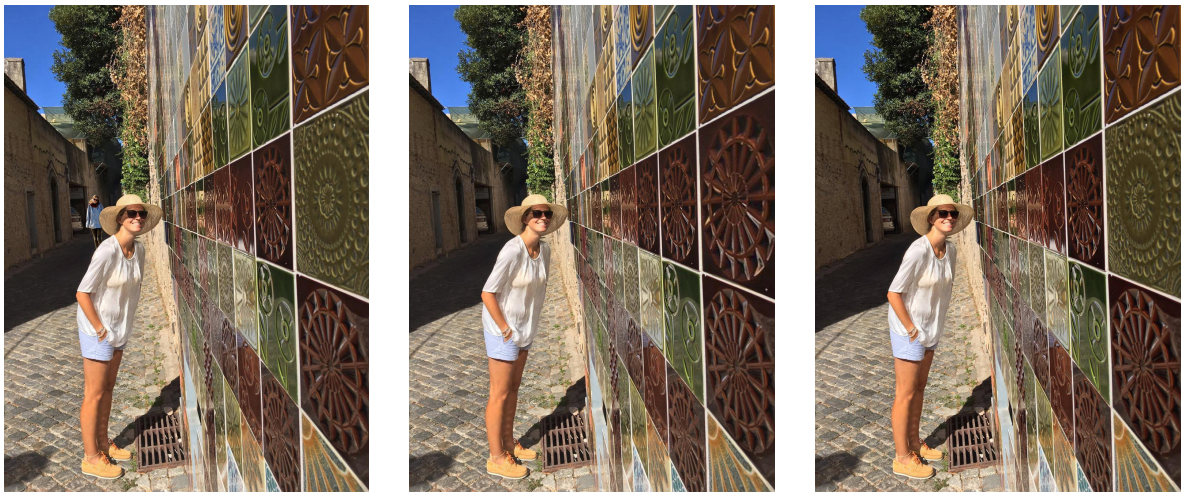
2.1.3 Falsification detection

Generally speaking, falsifications have usually a more notable effect on semantics of the image compared to manipulations. The spectrum of falsifications is very wide. Nowadays, the most publicized and feared by mainstream medias are falsifications with Generative Adversarial Networks (GANs), the so-called *deep fakes*. GAN is a recently developed type of neural network capable of generating data. However, this kind of data generation remains until now surprisingly easy to spot as long as they are generated by a CNN. In [Wan+19] authors are able with careful pre- and post-processing (of the datasets not during the generation of images) and data augmentation to create a "universal" detector. This detector is trained on only one specific generator of fake images (Pro-GAN) and provides good results and generalization for 10 others generators. Other generators have different architectures, training procedures and data. Some specific research works have also been conducted in Image Forensics field, on the detection of fake facial images produced with GAN [Rö+19], which achieves very good results. [Rö+19] proposes in addition a *few-shots learning* framework to mitigate drop of performance on unseen datasets during training. Therefore, it appears reasonable that for now we do not consider the detection of deep fakes as a priority of this thesis work.

Beside that, the prevalent falsified images in the wild are created with the help of software such as Photoshop or GIMP. There are several categories of falsifications, the three most common ones are depicted in Figure 2.3.

When a part of an image is copied and then pasted in a different image, the falsification is called *splicing*. With *copy-move*, a part of an image is copied and pasted within the same image, so both parts may share similar statistics and development process, especially when the copied area has not been subject to manipulations, *e.g.* scaling. *Inpainting* defines an operation where a part of the image is erased and often replaced by a textured pattern. Inpainting can be of two types: diffusion-based or copy-move based. With the diffusion approach, a local diffusion model of the image is exploited to extrapolate pixel values on the zone to be forged. In the copy-move based approach, small patches in the falsified region are replaced by other patches with similar neighbors. It is different from copy-move as the size of considered areas is much smaller, so distance between clones is usually also smaller and number of clones is much higher.

To tackle these kinds of forgery, there exist mostly specific detectors but also general ones.



(a) (b) (c)

Figure 2.3 – Examples of the most common categories of falsifications. (a) Splicing (another character has been added in the background); (b) Copy-move (same faience has been copied on another place); (c) Inpainting (a door in the background has been removed).

To detect copy-move specifically, the classical approaches try to match features of parts of the image. Areas considered are either exhaustive (often combined with an efficient searching algorithm [CPV15a]) or only points of interest [Yan+19]. Recently, a deep-learning based approach [WAAN18] has been proposed, which also aims to match similar features. More details about these methods are provided in Chapter 5.

Strategies to detect splicing and inpainting share a common intuition: forged part should deviate from pristine part of the image in terms of statistics or development process. One interesting and well-known method is SpliceBuster [CPV15b]. A local Gaussian model, based on patches, is learned for the background. Another model with uniform probability density function is learned for the suspicious zone. Based on distance to each model, the patches of the suspicious part are predicted as pristine or forged. This process produces a heatmap for the localization. Some recent methods have kept the same idea: detecting deviation from a reference model. For instance in [Huh+18], a model is learned to predict `EXIF` tags, metadata, of images. During a test phase, the model should predict consistent `EXIF` tags for every crops of the image. If the image has undergone a splicing, some crops would exhibit other `EXIF` tags. A score of self-consistency of the predicted `EXIF` tag is computed to generate a heatmap of suspicious region.

Recently and quite similarly, authors of [CV20] and [MS20] have proposed to use a specific type of neural network called *Siamese network*, for image forensics tasks. Traditionally, a

Siamese network is used to learn a similarity/dissimilarity metric between pairs of samples. It is composed of two parts:

— The first part with two branches and shared weights is used to extract features;

— A second part, which takes the two sets of extracted features as input, and then computes a similarity/dissimilarity score: *e.g.*, 0 if input samples are very similar, and 1 otherwise.

The learning by pair allows the use of smaller databases compared to classical learning, as a larger number of paired images can be produced even from a relatively small dataset. In image forensics, the learned similarity, or dissimilarity metric, represents the *forensic traces* [MS20]. For [CV20], the forensic trace is related to the camera model. Two images from a same camera model should have a score of 0, otherwise a score of 1. Therefore, the forensic trace can be used to build a camera model detector. In [MS20], the authors also consider manipulation operations and their parameters, in addition to the camera model, in order to create training pairs of images with similar or different forensic traces.

A common issue with machine learning and computer vision is the *generalization* to new datasets. Usually performance decreases on a testing dataset that is too different from the one used for training. It is also usually difficult to detect unseen classes during training, even generated from the same dataset. For instance, it would usually require a re-training for a median filtering detector to be able to well detect Gaussian blurring, even on the same dataset. In 2018, the authors of the pre-print [Coz+18] proposed a new neural network to overcome these limitations. It is an *auto-encoder* with a special loss on the latent space. An auto-encoder is a neural network composed of two parts. The first one, called the *encoder*, encodes the input into a *latent space*. Usually this latent space is of smaller dimension compared to input dimension. Then the second part, the *decoder*, uses the encoded representation of the input to reconstruct it. It is typically trained to minimize the reconstruction error. Auto-encoders are commonly used as dimensionality reduction methods. In [Coz+18], the representations in the latent space are used for classification of input samples between *fake* and *real* images. During the training, a loss in the latent space is computed in addition to the reconstruction error. Latent space is in $\mathbb{R}^{2N}$. An activation in the first N dimensions would indicate one class, for example *fake*, and activation in the last N dimensions would indicate the other class, for example *original*. During training a new loss is computed to enforce such behavior in the latent space. Combination of the two losses results in a better generalization capability and more efficient *few shots learning*. Few shots learning is the ability for a classifier to be able to classify a new class with only few labelled new examples and lightweight fine-tuning.

We have investigated similar ideas independently almost at the same time. We use *Variational Auto-Encoder* (VAE) [KW14] instead of a vanilla auto-encoder. In VAE, the latent space

is constrained to have Gaussian distribution. Thus, representations are normally more concentrated in the latent space than with the regular auto-encoders. In our work VAE is fed with local patches, instead of full-sized images as in [Coz+18]. Following roughly the idea of PixelCNN++ [Sal+17], instead of reconstruction of the input sample the network is intended to reconstruct a neighboring patch. Intuition is that it would be easy to reconstruct neighboring patch in an original image as natural images are quite regular. An anomaly, falsification or manipulation, would introduce difficulties in reconstructing neighboring patch. Idea is then to use either reconstruction error or latent variables for classification. Use of VAE and its constraint on latent space would allow a good generalization capability. At the same time this is an unsupervised training as no label is required, only a database containing original images is used for training. Results were promising. Unfortunately it would have required more investigations on the architecture to obtain a network that converges easily, and we had other tasks and constraints at that time which forced us to put aside this piece of work. It will be interesting in the future to conduct further studies in this promising research line.

The authors of [Bap+19] proposed an interesting strategy still in this trend, with auto-encoders and the detection of abnormal transitions in the image. Their neural network is composed of two branches: one branch is a regular CNN-encoder and the other one is a *Long Short Term Memory* (LSTM) branch. LSTM is a neural network cell that handles sequential data. In their work Radon features are extracted from patches and fed sequentially, following a path in the image. Radon features are discriminative features for resampling detection. In the CNN-encoder branch, pixels are fed directly. Feature maps from these two branches are then concatenated to be fed in a CNN-decoder. It produces directly a heatmap of image size that is compared to ground-truth mask of falsification during training. In total, it makes a very large network and LSTM cells are especially heavy with a lot of parameters to train and hard to converge. Therefore they use synthetic data for training. Authors report impressive performance on NIST 16 [DAR17] and IEEE Forensics [CGV15] datasets. They also provide code for the method [1] and we have tested it on DEFALS database. The pre-trained version performed very poorly, probably due to the large difference between the synthetic data and the testing dataset. We tried to fine-tune it and re-train it on DEFALS data, but probably due to the complication caused by the size of the network, we did not succeed in obtaining satisfactory results. It is for us another observation relevant to our thesis objective, that is, from an operational point of view, methods are expected to be easily adaptable and flexible to new data, even though large neural networks are able to achieve impressive results on specific datasets.

1. https://github.com/jawadbappy/forgery_localization_HLED

2.1.4 Localization

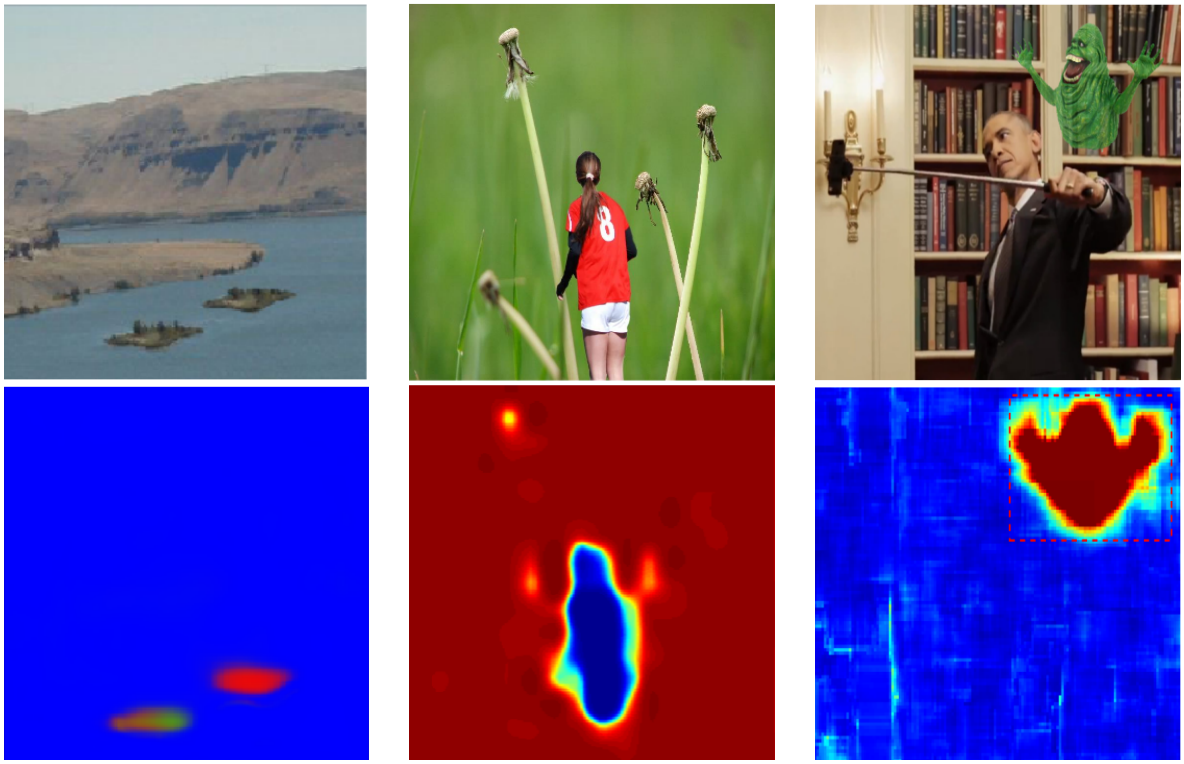
The literature of image falsification detection is interested in forgery localization, jointly with a binary global decision. Most of the methods introduced in the previous Section 2.1.3 also produce a heatmap of the suspicious part(s) of an image. The method based on patches could allow localization at patch or pixel level, though it is usually with a lot of false positives. Even a very low percentage of false positive and negative of 2% would become very noticeable in a 2000×3000 image divided in 8×8 patches with an overlap of 3 pixels, because the number of patches would be more than 600000 and would thus approximately contain 12000 errors. A post-processing is therefore necessary.

The difficulty to produce a global decision (at the image level) along with a local decision (a heatmap) mainly resides in the following observation: if falsification or manipulation is small compared to the size of the image, it could be indiscernible from the false positive. Authors of [Mar+20] combine the two as it first predicts a heatmap and then a global decision based on the heatmap. It only requires the global labels for the training. Some examples of localization and heatmaps can be found in Figure 2.4.

To summarize, localization is closely related to the detection of manipulation and falsification but this is not that straightforward. The most promising methods seem to be those producing a heatmap the size of the image, in an end-to-end manner, such as [WAAN18], [Mar+20], and [Bap+19]. Patch-based approaches should have in theory a good spatial accuracy, although the required post-processing in practice tends to reduce the final accuracy.

2.2 Available databases

To assess performance and compare methods, the information forensics community relies on public databases that we present here. These databases evolve following the progress of opponents and their software, and image editing software, though these databases are not meant to be perfectly realistic. Manipulations and falsifications in these datasets are often easier to detect than for images in the wild. It is mainly because images in the wild generally undergo a lot of post-processing that cover tampering fingerprints while public databases are only lightly post-processed. Though, it is assumed that being able to tackle efficiently easier problems of public database would then help to detect falsifications and manipulations in the wild. It is a standard methodology to first deal with smaller and easier problems before approaching more complex ones. Beside that, there exists so big a variety of image manipulations and falsifications regarding the source of images, the type of tampering, the



(a) (b) (c)

Figure 2.4 – Examples of localization and heatmaps: (a) From [WAAN18], where blue color indicates pristine, red color indicates target area of copy-move and green color is the source area; (b) From [Bap+19], where blue color indicates suspicious area(s) and red color indicates pristine areas; (c) From [CPV15b], where red color indicates suspicious area(s) and blue color indicates pristine areas.

image editing software and their versions, *etc.*, that choices must be made. For the time being, it is impossible to have a dataset that would cover all these diversities.

2.2.1 RAW images

For manipulation detection, datasets of `RAW` images are used. It is essential to control the development process as it will largely influence performance of manipulation detection. From the raw images, a development of choice is performed before adding a manipulation and then eventually a possible post-processing. The Dresden [GB10] database is probably the most popular database among the community. It contains 1491 images coming from 4 different cameras. The Dresden pictures are taken in various indoor and outdoor scenes with different lighting and exposure. This dataset can be enlarged with 16961 JPEG images coming

(a) (b) (c) (d)

Figure 2.5 – Examples of images from the Dresden database [GB10].

from 27 different cameras, still from the same scenes. These images are in high resolution, around 2000×3000 pixels. Some examples are shown in Figure 2.5. In the close field of steganalysis, there also exist two large databases of `RAW` images, respectively from the BOWS challenge [FB08] and from the ALASKA steganalysis challenge [CGB19]. There are 10000 original images in the BOWS challenge. Yet, the BOWS images are grayscale and resized to 256×256, which does not allow a total control on the development process. In the Alaska challenge, the dataset is composed of 50000 images from 21 cameras. The Alaska images are really diverse, with various sizes and resolutions.

2.2.2 Synthetic dataset for falsification detection

Recently, the COCO dataset [Lin+14] has gained interest for the development of synthetic datasets of falsified images. The COCO dataset has been primary developed for image segmentation problems. It contains 200000 images with semantic segmentation information, which makes it possible to automatically create semantically meaningful falsified images with copy-move or splicing forgeries. With the large number of images it contains, it is possible to create a lot of samples to pre-train a neural network. The authors of [WAAN18] and [Bap+19] have successfully used such a strategy with their networks. Some examples of automatically created copy-move forgeries from [WAAN18] can be found in Figure 2.6.

2.2.3 Manipulation and falsification databases

2.2.3.1 Public databases

To the best of our knowledge, the first public database for falsification detection is the one from Columbia University [NC04]. This dataset is quite small with 1845 grayscale images of

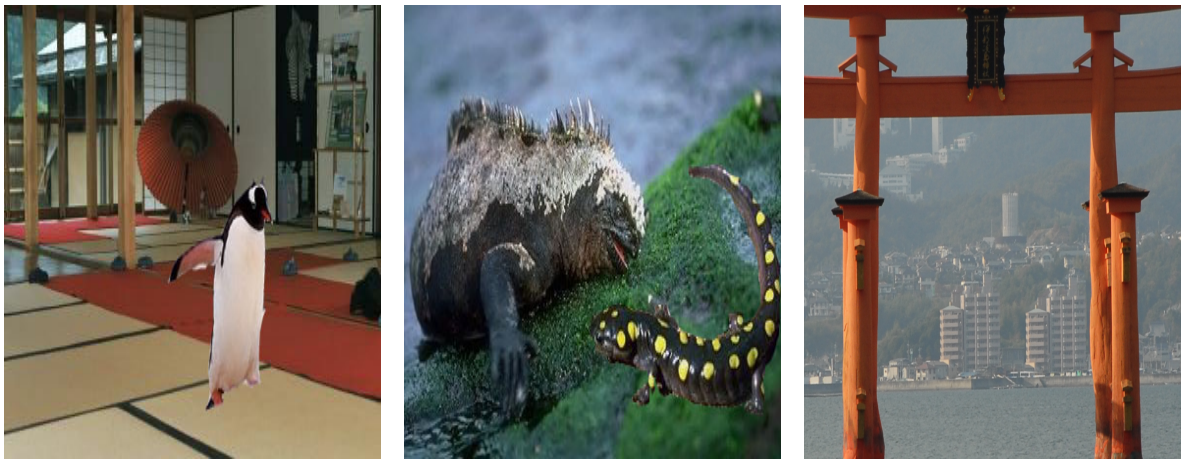
(a) (b) (c)

Figure 2.6 – Examples of automatically created copy-moved images by authors of [WAAN18].

size 128×128 pixels and 363 uncompressed images of bigger size (from 757×568 to 1152×768). Half of these images have been spliced without post-processing. Splicing falsifications are naive and performed on blocks.

Then, researchers from the Chinese Academy of Sciences have collected a database named CASIA Image Tampering Detection Evaluation (CASIA ITDE) [DWT13] in 2013. The tampered images were created with the help of Photoshop software. A first version, CASIA ITDE v1.0 is composed of basic splicing falsification only, on JPEG color images of fixed size of 384×256 without post-processing. In total there are 1721 images: 800 images are authentic and 921 are tampered. The version 2.0 of CASIA is larger, with 12323 color images of various sizes of at most 800×600 pixels (7200 authentic images and 5123 tampered). Some images are JPEG-compressed with various quality factors and some others are uncompressed (`BMP` or `TIFF` formats). The falsifications are visually more realistic in version 2.0. There are also copy-move images in addition to spliced images in CASIA ITDE v2.0. Some images are post-processed with a blurring operation to cover falsification fingerprints.

In 2013, the IEEE IFS-TC Image Forensics Challenge [CGV15] has provided another database. The challenge had two phases: a first phase of falsification detection at the image level and a second one of falsification localization. Notably, a leak in meta-data [GKR13] had allowed a 100% of correct identification at the image level. In the training set, there are 450 tampered images with the corresponding masks of falsifications and 1050 pristine images. There are also 5076 test images. The images are of various sizes, mostly of high resolution (around



(a) (b) (c)

Figure 2.7 – Example of tampered images from public databases. (a) Image from CASIA ITDE v1.0 [DWT13], the penguin has been spliced. (b) Image from CASIA ITDE v2.0 [DWT13], the salamander has been spliced. (c) Image from GRIP CMFD database [CPV15a], big building in the background scene has been copy-moved.

2000×3000 pixels) and from 25 cameras.

In 2017, the U.S. government DARPA released a quite large corpus at the occasion of the 2017 Nimble Challenge [DAR17]. The challenge was about:

— Manipulation detection and localization;
— Splicing detection and localization;
— Provenance filtering (identification of the source);
— End-to-end provenance graph.

The dataset for Manipulation detection and localization is composed of a total of 10000 training images and 1083 videos.

At last, GRIP CMFD [CPV15a] and CoMoFoD [Tra+13] are two small databases specifically designed for copy-move detection. GRIP CMFD is composed of 80 `PNG` images of size 1024×768. CoMoFoD database contains 260 images, 200 small ones of 512×512 pixels and 60 bigger ones of 3000×2000 pixels. On these 260 images several post-processing operations have been applied (JPEG compression, addition of noise, blurring, brightness change, color reduction and color adjustments). It makes a dataset of 10000 images (with only 260 distinct images) to test robustness of copy-move detectors against post-processing.

Some examples of falsified images are illustrated in Figure 2.7.

2.2.3.2 DEFALS database

For the DEFALS challenge, three databases have been provided to participants, one for each stage. These databases are private and cannot be shared. Tampered images have undergone falsifications that are divided in five categories:

1. Substitution of a small object by a patter in image;
2. Insertion of an object or large area in image;
3. Deletion of an object or large area in image;
4. Tampering of characters or number (license plate, street name, ...);
5. Camouflage (hiding of a part of an object in the scene).

Basically, operations 1., 3. and 5. are inpainting. operations 2. can be either a splicing or a copy-move and operation 4. is mainly small copy-move of figures or letters. Some images have undergone several falsifications, all of very good quality and performed by trained professionals using recent software. In some images, it is hard to detect the tampering by the naked eyes. In addition to these falsifications, some images also have undergone manipulations: blurring, cropping, resizing, color enhancement, *etc*. Images with only manipulations are considered as authentic in the context of the contest. Images are coming from various sources (cameras, smartphones, satellite images, *etc.*) and mainly in large size. The scenes are very diverse (outdoor, indoor and also photographic studios). The three databases are of very good quality both for the realism of falsifications and the diversity in sources and scenes.

The first stage of the contest happened in May 2018 and was actually a preliminary stage, to test the contest infrastructure and processes. The training dataset had 650 images with `XML` files describing the manipulations and falsifications. The objective was binary classification between falsified or not on a testing set. The test images were copies of previously released training images. We used the image hash and image size on disk to match the test images with the training ones, and we used the training labels to carry out classification. There was also a meta-data leak as the `EXIF` data had not been erased. It was possible to read in the `EXIF` data whether the image had been falsified or not. Although we did not use it as we already had achieved a perfect classification using matching with the training images.

The second stage took place in February 2019. It was the first official stage. The goal was also binary classification. This time, meta-data had been erased but because of the pipeline of falsifications, all falsified images had a chroma subsampling pattern of 4:4:4, while original images had different patterns, including 4:4:4 but also 4:4:2 and 4:2:0. It allowed us to extract a sub-list of suspicious images with a 4:4:4 pattern to classify. In addition, because of the falsification pipeline, the color dynamics were different between authentic and falsified

images, which can even be spotted by the naked eyes. Thus, this made the classification easier and we were able to achieve F1-score higher than 90% on the testing set.

For the last stage, the organizing team provided 2654 images with their corresponding masks of falsification as the training set. The objective of this stage was both binary classification and falsification localization. The two F1-scores (image-level and pixel-level) were computed. This time, color dynamics and chroma subsampling patterns were mostly coherent between the authentic and the falsified images. However, a leak was present on the image sizes. We have noticed some specific and dominant image sizes among the copy-move images identified by a reference detector. We then made an assumption that all the images with these specific sizes in the training set were falsified. We have verified this assumption during the training phase with a submission that indeed reached an F1-score very close to 1 at the image level. It allowed us to extract an almost complete list of falsified images to focus on, so our image-level F1-score was close to 100% during the testing phase. Localization was much harder because of the mix of falsifications: copy-move, splicing and inpainting. Only copy-move detectors gave us satisfactory results.

3 | Background knowledge in Machine Learning

Contents

To tackle Image Forensics problems defined previously, most of the existing detectors use tools that fall in the broad category of *Machine Learning* tools. Machine learning is a sub-field of artificial intelligence. It studies algorithms that make use of *training data*, *i.e.* a set of examples with/without labels, to produce decision or prediction for some other samples, *i.e.* the test samples. Usually the prediction is done on labels of test samples but it could also be to clusterize data without training labels. Contrary to conventional algorithms, machine learning algorithms are not explicitly programmed to do so. A simple example is the linear regression in 2D: a straight line is fitted on some samples which allows us to predict and extrapolate behaviour pattern of the data outside of these samples. The fitting is performed automatically through optimization of a chosen criterion, coefficients of the linear regression are not explicitly chosen by the programmer. Another simple machine model is the *nearest neighbor method*: characteristics of data are extrapolated based on similarity with known samples. With nearest neighbor method, the algorithm does not hard code the output but makes it dependent on the data. However, algorithms based on conditional statements, *i.e.* `if-else-then` are not considered as machine learning as output is explicitly programmed.

3.1 Definitions

Usually in machine learning, X is defined as the data or the *input*. In our case, it is the images or the patches extracted from images, therefore $X \in \mathbb{R}^{n \times p}$ with n the number of samples and p the dimensionality of the images or patches. It could also be of other nature in other context: measurements of some physical signal, or descriptors of an object such as length and width of leaves, or even categorical (qualitative) such as blood type A, B, AB or O. Meanwhile y represents the *label* or the output. It could be either a class, *e.g.* cat or dog, or a quantitative value, *e.g.* stock market value on the following day. In our case it is the class: pristine or tampered/falsified/manipulated. The inputs affect the outputs: a joint law of probability for (X, y) is supposed.

3.1.1 Supervised, semi-supervised and unsupervised learning

The samples in X are then divided in two sets:

1. the *training set*, which will be used to train algorithms, and
2. the *testing set*, which is the set for which we should predict class or value.

In our case, we have some unknown images that we would like to test and find out if they are pristine and not: they correspond to the testing set. The known images correspond to the

training set.

In the case of *supervised learning*, the *labels* (*e.g. class or values*) are known on the training set and of course unknown on testing set. In the *unsupervised learning* setting, no labels are known on both the training and the testing sets. The *semi-supervised* scenario corresponds to the availability of some labels on the training set in addition to samples with unknown labels. One also defines the *weakly-supervised* setting as a subset of semi-supervised learning where very few labeled samples are available.

In supervised and semi-supervised learning, a third set is usually defined: the *validation set*. It is composed of samples extracted from the training set. A crucial point is that no leak is permitted between the three sets: training, validation and testing. The validation set is used to give an estimation of the performance of the algorithm on unknown testing set. Of course, this performance estimator exhibits a high variance, thus a *cross-validation* step is usually performed. During the cross-validation, the training set is divided into K folds. $K-1$ folds are used as training set while the remaining one is used as validation set to estimate performance. This operation is repeated K times with rotation of the validation set. It produces K estimations of the performance.

There also exists *online learning* where samples are supplied sequentially to the learning algorithm. An example is data coming from sensors. The ordering of the measurements is important as it reflects an evolution over time. Due to the high volume of data in time it may not be possible to store measurements. Therefore samples are supplied sequentially to the model to update it. Finally, in *reinforcement learning* an *agent* interacts with an *environment*. The aim is to learn optimal behavior for the agent regarding a reward function. It is usually not possible to describe exhaustively all the rewards associated to every possible chain of agent actions in the environment. Therefore the framework does not use a couple of X data and y labels as in supervised and semi-supervised learning. Instead, the environment and rewards are explored by the agent which uses at the same time this partial knowledge to extrapolate optimal chain of actions. It usually relies on a physical simulation system. These two types of learning, *i.e.*, online learning and reinforcement learning, will not be discussed further as at present they do not apply directly to Image Forensics problems.

3.1.2 Classification and regression

Classification aims at predicting qualitative values, such as digit recognition or content-based image classification. Regression algorithms output quantitative values such as salary or house prices. The support of regression output is usually continuous and real valued. Both

approaches, in the case of supervised or semi-supervised learning, aim at finding a mapping function f such as $y = f(X)$. In the case of unsupervised learning it is a little different as usually the objective is to find an underlying structure of the data that would allow us to describe or cluster it. There are no explicit labels (*i.e.* y values).

3.2 Statistical learning

Until 2012 and the renewed interest for neural networks methods, algorithms based on statistical learning were the state of the art. Statistical learning is derived using tools from the statistical field. It makes the assumption that there is an unknown *probability distribution* over the couple (X, y). It aims at finding a function f such that $f(X) \sim y$ (*i.e.* $f(X)$ has the same distribution as y). A probability distribution is a function that describes the likelihood of each event of a random phenomenon. In our case, it means finding probability of each specific couple (X_i, y_j). Usually, (X, y) is continuous so f is a continuous function as well. A good overview for these methods can be found in [HTF01] and a more specific overview oriented towards computer vision applications can be found in [Bis06]. Here, we will only briefly present some existing methods.

The different methods of statistical learning suppose different type of function f to look for. For instance, linear models search for a linear function f to fit training data. There also exists methods based on decision trees, so the space of f is the space of decision tree. To find the best candidate in the function space, we need a *loss function* L that would penalize functions providing false predictions and in the meanwhile promote functions offering good predictions. The objective is then to find f^* such as:

$$f^* = \underset{f}{\operatorname{argmin}} \underset{(x,y)}{\mathbb{E}} L\left(y, f(X)\right). \tag{3.1}$$

The historical and most powerful methods are linear regression, logistic regression, kernel methods (SVMs, for support vector machines), the ones based on trees such as Random Forest or Gradient Boosting, nearest-neighbors (k-means and k-nearest neighbors). We will not give further explanations on these methods as they are not used in this manuscript.

3.2.1 Gaussian Mixture Model

The Gaussian Mixture Model (GMM) is a statistical and *generative* model. A *generative* means that, based on some assumptions, it is the ideal data-generation process of some random variable that it is looked for. We use GMM in the subsequent chapters to model how

small patches are generated. Then, the random variable to model is a random vector of the pixels, the constituents of a patch.

GMM is based on the most important law of probability in statistics: the Gaussian distribution. Let $X = (X_1, \ldots, X_p) \in \mathbb{R}^p$ be a continuous random vector, the probability density function f of a Gaussian distribution is:

$$f_X(x) = \frac{1}{(2\pi)^p \sqrt{\det(\Sigma)}} \exp\left(-\frac{1}{2}(x-\mu)^t \Sigma^{-1}(x-\mu)\right), x \in \mathbb{R}^p, \tag{3.2}$$

with Σ the covariance matrix of size $p \times p$ which is symmetric and positive definite, and μ the mean vector of size p. $X \sim \mathcal{N}(\mu, \Sigma)$ means that random vector X follows a Gaussian distribution of parameters μ and Σ. This distribution is very simple and uni-modal (with a unique peak).

To approach more complex distributions, for instance distributions with multiple modes, mixture models are used. Mixture models do not require individual identification of the nodes. It is a weighted sum of simple probability densities. The weights sum to 1, so that the combination is convex and the mixture has a valid probability density function. The probability density function of a Gaussian Mixture Model with K components is thus defined as:

$$f_X(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}\left(x; \mu_k, \Sigma_k\right), x \in \mathbb{R}^p, \tag{3.3}$$

with π_k the weights (non-negative values summing up to 1), and μ_k and Σ_k parameters of the K Gaussian components. These models are universal approximator, which means that they can approximate exactly any density when $N \to \infty$.

Parameters of the GMM are usually learned with the Expectation-Maximization (EM) procedure. Let $\Theta = (\theta_1, \ldots, \theta_N)$ be a collection of N observations that we want to approximate with a GMM. The aim is to maximize the log-likelihood of the model for the dataset Θ:

$$\ln(p(\Theta|\boldsymbol{\mu}, \boldsymbol{\Sigma})) = \sum_{l=1}^{N} \ln\left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\theta_l; \mu_k, \Sigma_k)\right), \tag{3.4}$$

with $\boldsymbol{\mu} = \{\mu_1, \ldots, \mu_N\}$ and $\boldsymbol{\Sigma} = \{\Sigma_1, \ldots, \Sigma_N\}$.
First we compute the derivative of this log-likelihood with respect to the mean μ_k:

$$\frac{\partial \ln(p(\Theta|\boldsymbol{\mu}, \boldsymbol{\Sigma}))}{\partial \mu_k} = -\left(\sum_{l=1}^{N} \frac{\pi_k \mathcal{N}(\theta_l; \mu_k, \Sigma_k)}{\sum_{s=1}^{N} \pi_s \mathcal{N}(\theta_l; \mu_s, \Sigma_s)}\right) \Sigma_k^{-1}(\theta_l - \mu_k). \tag{3.5}$$

The term after the sum sign inside the parenthesis is named *responsibility*. Responsibilities are latent variables r_{jk} such that $r_{jk} = 1$ if the component k has generated the sample θ_j (and

0 otherwise). It can be shown using Bayes theorem that:

$$P(k|\theta_l) = \frac{P(k)P(\theta_l|k)}{P(\theta_l)} = \frac{\pi_k \mathcal{N}(\theta_l; \mu_k, \Sigma_k)}{\sum_{s=1}^{N} \pi_s \mathcal{N}(\theta_l; \mu_s, \Sigma_s)} = r_{lk}. \tag{3.6}$$

These very same responsibilities also appear in the partial derivative of the log-likelihood with respect to the weights π_k or the covariance matrices Σ_k. Therefore, to be able to *maximize* log-likelihood and find optimal values for π_k, μ_k and Σ_k, a first step of calculating responsibilities has to be performed. This step is called *Expectation* step (E step). During this E step π_k, μ_k and Σ_k are assumed to be known. During the M step (*Maximization* step), responsibilities are assumed to be known. Hence, these two steps are performed alternatively until reaching a convergence criterion. This criterion is usually based on a minimum threshold for improvement of the log-likelihood. Main limitation of this EM algorithm is that it is very sensitive to the initialization of π_k, μ_k and Σ_k. Initialization can be performed either randomly or with the help of an auxiliary algorithm such as k-means. In both cases, multiple initializations are performed to find and keep only the one with the highest log-likelihood.

Another limitation of GMM is that it contains a large number of parameters. We have for each covariance matrix, $p \times (p+1)/2$ independent parameters, p parameters for each mean vector and K weights, so a total of $K \times \left(\frac{1}{2}p^2 + \frac{3}{2}p + 1\right)$ independent parameters. Thus, to reduce this number, a common trick is to enforce a particular structure to covariance matrices (diagonal, spherical, ...) with as trade-off on the loss in the descriptive power of the model. It also helps to speed-up the computation of the Maximization step, as this step requires the inversion of the covariance matrices.

3.3 Neural networks

Neural networks (NN) are biological inspired models. They are composed of *layers*. The first layer receives the input of the model, while the last layer outputs a classification or regression depending on the problem. Between the two there are *hidden* layers in cascade. These layers are composed of *neurons*. A neuron loosely models the neuron of the biological brain by reacting to a received signal and outputting a modified version for the following neuron. There are connections between neurons and between layers. A network is called *fully-connected* if each neuron is connected to all neurons of the following layer. A graphical illustration of these principles can be found in Figure 3.1.

A neuron is typically composed of weights, bias — just as a linear regression model — and an activation function:

$$x_{i,j} = f_{i,j}(\boldsymbol{x_{j-1}}) = \sigma(\boldsymbol{w_{i,j}^T x_{j-1}} + \boldsymbol{b_{i,j}}), \tag{3.7}$$

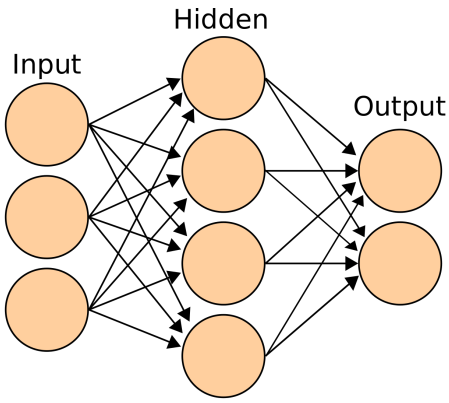


Figure 3.1 – A fully-connected neural network for binary classification, with one hidden layer of size 4 and inputs $\in \mathbb{R}^3$. Orange dots are neurons and arrows symbolize connections. Figure from Wikipedia (https://commons.wikimedia.org/w/index.php?curid=1496812).

where $\boldsymbol{x_{j-1}}$ are the inputs (a vector) from a previous and connected layer $j-1$, $x_{i,j}$ is the output of the neuron i in layer j, σ is the activation function, $\boldsymbol{w_{i,j}}$ are the weights of neuron i in layer j (a vector) and $\boldsymbol{b_{i,j}}$ is the bias of neuron i in layer j (a vector). The $x_{i,j}$ from all neurons i in layer j then form $\boldsymbol{x_j}$ the input vector for the layer $j+1$ neurons. A typical activation function is the Rectified Linear Unit (ReLU):

$$\mathrm{ReLU}(x) = \begin{cases} 0, & \text{if } x < 0 \\ x, & \text{otherwise.} \end{cases} \tag{3.8}$$

George Cybenko showed in 1989 [Cyb89] that a NN with a single hidden layer of arbitrary size and with sigmoid activation can approximate any continuous and real-valued functions on a compact subset of $\mathbb{R}^n$. However, this work does not provide procedure to determine the number of weights and neither there values.

The most used procedure to find optimal parameters for the neurons is called *back-propagation*. Training examples are fed to the network, and an error loss is computed between prediction and ground-truth. For example, with a classification and cross-entropy loss:

$$R(\Theta) = -\frac{1}{N}\sum_{l=1}^{N}\sum_{k=1}^{K} y_{l,k}\log(f_{k,M}(f_{M-1}(\dots f_1(\boldsymbol{x_0^l})))), \tag{3.9}$$

where Θ are the parameters for all layers, N is the number of samples, K is the number of neurons in last layer which also corresponds to the number of classes, M is the number of layers, $y_{l,k}$ is the one-hot encoded ground-truth, $\boldsymbol{x_0^l}$ is the l-th input sample, $f_{k,M}$ the function with activation for the neuron k in the last layer M, and (with a slight abuse of notation) $f_{M-1}, \dots, f_1$ represent functions at other layers which output a vector $\boldsymbol{x_{M-1}^l}, \dots, \boldsymbol{x_1^l}$.

The error is first computed for the last output layer M, and partial derivatives are computed with respect to the parameters of neurons in this layer: $\boldsymbol{w_{k,M}}$ and $\boldsymbol{b_{k,M}}$, to find optimal parameters. Gradient descent is used as optimization method. Then, the same computation is performed for the second-last layer. This time $f_{k,M}$ is expressed with $f_{i,M-1}$ the function of the second-last layer with $i = [1, \ldots, K']$ (K' being the number of neurons in second-last layer), to allow the computation of partial derivatives with regard to $\boldsymbol{w_{i,M-1}}$ and $\boldsymbol{b_{i,M-1}}$. The computation continues for other layers and accordingly the error is back-propagated in all the network until it reaches the input layer.

Usually, neural networks stack several hidden layers (more than 3 can be considered deep-learning) with a large number of neurons. Hence, the number of parameters to optimize is quite large. Therefore, a large number of examples is required to train the network. It becomes then impossible in terms of computing power and training time to compute errors for all examples at the same time. So, instead of the computation of gradients following every direction, *i.e.* using every sample, only a random, *stochastic*, subset of examples are selected each time. They are called a *batch*. Gradient descent is performed sequentially on all batches. The batch size and learning rate of the gradient descent of this *Stochastic Gradient Descent* (SGD) are hyper-parameters to be defined. Another advantage of introducing randomness with the SGD is that it will help the network converge. The optimization problem is generally not convex, with probably several local minima and the dimensionality of the space is very large. Classical optimization methods without randomness fail in this setting, and proximal operators are not applicable in this setting because of their computation cost. When all batches are passed through the network, it is called an *epoch* — and then new random batches are drawn. Networks are trained on several epochs. The number of epochs is also a hyper-parameter.

Some more advanced SGD methods have then been developed to speedup convergence such as Adam [KB15]. Generally, raw values (without features extraction) are fed to the network. Features are supposed to be constructed in the hidden layers of the model and classified by the output layer. A good overview of neural network and deep learning methods can be found in [GBC16].

3.3.1 Convolutional Neural Network (CNN) classifiers

We have presented in Section 3.3 regular neural networks which deal with vectors as input. Images are however more conveniently represented as matrices. The matrices could be vectorized but this would lead to some loss of spatial information. Moreover, it would require too many parameters with the fully-connected neurons. Therefore, in order to deal

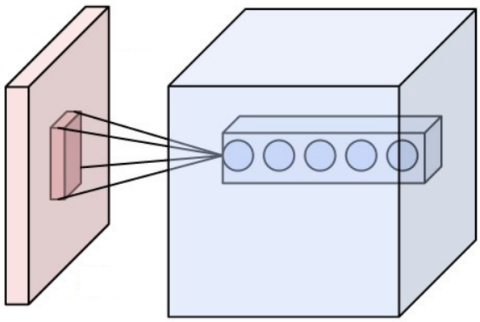


Figure 3.2 – An example of a convolutional layer. The convolutional kernels are in blue and of size k, k, l. Blue dots are individual weights of the convolutional kernels that have the same *receptive field* (the small red square), *i.e.* connections with values in input (big red square). Figure from Wikipedia (https://commons.wikimedia.org/w/index.php?curid=45659236).

with matrices, the weights $\boldsymbol{w_{i,j}}$ and biases $\boldsymbol{b_{i,j}}$ from Equation 3.7 are now matrices. The weights are not multiplied anymore but convolved and are called *convolutional kernels*:

$$\boldsymbol{x_{i,j}} = \sigma(\boldsymbol{w_{i,j}} * \boldsymbol{x_{j-1}} + \boldsymbol{b_{i,j}}). \tag{3.10}$$

An illustration of a convolutional layer can be found in Figure 3.2. Instead of neurons, a layer contains several convolution kernels. The spatial sizes of the kernels are hyper-parameters, usually between 3 and 9. After each layer, feature maps are produced to be fed to kernels of the next layer. So kernels of layer j are of size: k, k, l, with k being the spatial size and l being the number of kernels (which is usually higher than the number of feature maps in the previous layer to increase size). The horizontal and vertical sizes of the outputted feature maps depend on other hyper-parameters related to the convolution kernel which are the *stride*, *i.e.* how many pixels are shifted between two local operations of convolution, and the *padding*, *i.e.* how the borders are handled. Using the same kernel on the whole image reduces the number of parameters compared to a fully-connected network, which also reflects the assumption of translation invariance of natural images.

The recent hype for the deep learning methods is closely related to these CNNs as they perform extremely well on images, and often better than traditional methods. It is illustrated by the very large progress made in an image recognition challenge called ImageNet since 2012, when first CNN methods were applied. ImageNet is a very large dataset of 256×256 images with 1000 categories describing content of the image: cat, dog, car, *etc*. The top-5 accuracy, meaning that the actual category of the image is among the 5 most probable categories predicted by a classifier, has jumped from around 70% with traditional methods to around 90% with the latest deep learning methods.

Neural networks are an old idea and CNNs have been first proposed by Kunihiko Fukushima in 1980 [Fuk80]. However, training procedures were complicated and often included hand-crafted parts. In 1989, Yann Le Cun proposed a practical training method called *back-propagation* [LeC+89] that has been used ever since. These CNNs methods became successful only in 2012, as the required computation power was unavailable until then to apply CNNs on large-scale problems. Progress in *Graphics Processing Units* (GPUs) programming, on which convolutional operations are performed very quickly, are a major explanation for the late success of the CNNs.

The main drawback of CNN and NN in general is that the decision produced is hardly explainable. All these neurons and connections act somehow as a black box. With hand-crafted features and logistic regression, it is possible to extract the most important factors that lead to the decision. It is especially important in the health domain. Beside that, from a theoretical point view, there is, at the time, no guarantee of convergence for the training procedure whatsoever. The only guarantee that we have is that it produces very good empirical results. In addition, the training of CNNs is long and costly, as it requires large GPUs. There are no real guidelines on how to choose the architecture or the hyper-parameters. It is mostly done by trial and error. We can draw here a parallel [1] between drug discovery and neural network development. The chemistry and biology behind drugs that we use are so complex that we only have a limited understanding of it. To develop a new treatment, we have a few intuitions thanks to this limited knowledge, but development is mostly performed through trials, first on cells and then on animals and finally on humans. A drug is considered satisfactory when clinical trials expose large benefits (true positive rate) and very limited side-effects (false positive rate). Then the drug is sold and used by the layman. It is not so different for neural networks.

1. See https://www.youtube.com/watch?v=gG5NCkMerHU (last accessed: Sept. 2020).

4 | Model Adaptation

Contents

4.1 Motivation

4.1.1 Image manipulation detection

The starting point of this chapter is depicted in Figure 4.1 and Figure 4.2. Two classifiers, one based on GMMs and another CNN-based, are trained on patches of full-sized images to classify between manipulated and original patches (details of the classification pipelines are given in [FWC15]; [BS16] and later in Section 4.3.2 and Section 4.3.1). The manipulations considered are summarized in Table 4.1. The classifier is then used on a testing set that has undergone resizing (with bi-cubic interpolation) as a **pre-processing** operation, prior to adding a manipulation. It is here different from the more traditional studies about robustness against post-processing, as pre-processing does not alter manipulation fingerprints. Drops in accuracy are observed for several resizing factors. Intuitively, pixel statistics and dependencies between neighboring pixels are altered as images get smaller with downscaling (or bigger with upscaling). For example, downscaling usually induces sharper transitions as fewer pixels contribute to each transition. One can notice that the decrease in performance of GMMs is almost linear with the factor of resizing.

We denote *source* the training data of original size. Half of these data have undergone a manipulation.*Target* denotes the testing data that have undergone resizing as a pre-processing operation. Then, just like for the training set, half of these testing data are manipulated. Our objective is to distinguish between original *vs.* manipulated data. Ideally, accuracy should not drop because of pre-resizing. We shall first study how well a classifier trained on source data performs on target data, and secondly we seek a method to enhance performances on target data without re-training the detector from scratch. From an operational point of view, and in contrast to the usual approaches, we intend to provide a procedure that is flexible, adaptable regarding target sets, requires no labels (unsupervised setting) or very few labels

Table 4.1 – List of basic image manipulation operations to be detected.

ORI	No image modification
GF	Gaussian filtering with 3×3 kernel and $\sigma = 0.5$
MF	Median filtering with 3×3 kernel
USM	Unsharp masking with window size 3×3, and parameter 0.5 for the Laplacian filter to generate the sharpening filter kernel
WGN	White Gaussian noise addition with $\sigma = 2$
JPEG	JPEG compression with $Q = 90$

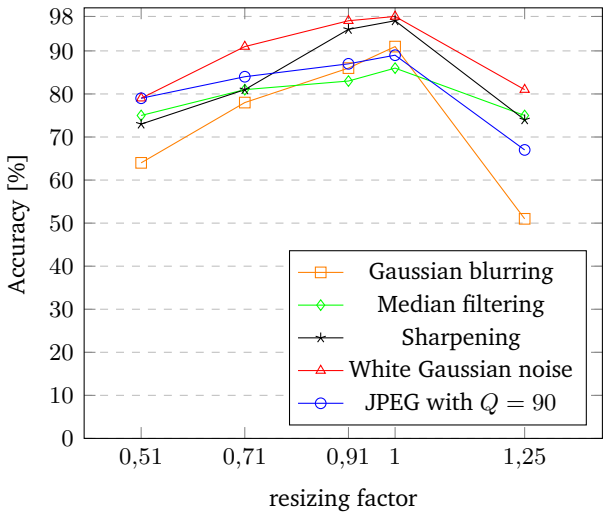


Figure 4.1 – Accuracy of GMM-based method under different resizing factors (bi-cubic interpolation) for several manipulations.

(weakly-supervised setting) on the target and is resource-friendly. Hence, our goal may be stated as being able to use already existing methods in new scenarios (such as pre-processing) or on new datasets, with only a slight adaptation yielding the same performance level. While with most detectors great care is taken regarding robustness to post-processing, it is usually not the case with pre-processing, which motivates our work.

The same kind of performance drop is also observed using the CNN classifier of Bayar and Stamm [BS16], as shown in Figure 4.2. Details of the method are described in Section 4.3.1).

To cope with these performance losses, it is possible to operate either on the model or on the features. For example, re-training the model/classifier from scratch on the target data is a form of model adaptation, though not practical because it is resource-hungry. Implicit models often have parameters that are not directly related to the data (*e.g.* quantization levels or filters' shape for SPAM [PBF10] and SRM [FK12]). Therefore, it is not straightforward to find a model adaptation procedure, especially in an unsupervised framework. In this case, the classical strategy instead is to adapt the features produced by implicit models. On the other hand, with explicit models such as GMM, it is quite direct to adapt parameters of the models using target data (like the covariance matrix of the data).

In this chapter, we focus on done on bi-cubic interpolation over 4×4-pixel neighborhoods for resizing, as we found it is the hardest case, compared to nearest-neighbor interpolation, linear interpolation or Lanczos interpolation over 8×8-pixel neighborhoods. Bi-cubic interpolation is also the most commonly used method, as it produces less visual artifacts than

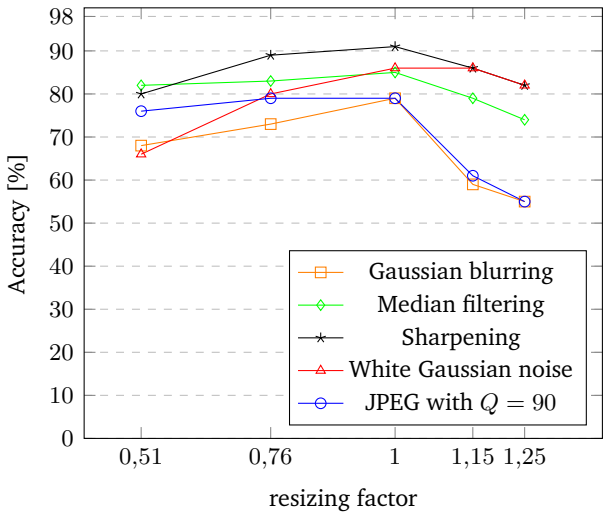


Figure 4.2 – Testing accuracy (in %) without any adaptation for CNN-based method [BS16].

nearest-neighbor interpolation.

4.1.2 Related problem in image recognition

In 2019, the authors of [AW19] have raised some quite related concerns about performance losses of CNN when exposed to pre-processing for image recognition. They claim that *" small translations or rescalings of the input image can drastically change the network's prediction."* For one experiment, they randomly draw 1000 images from ImageNet. For each image a square (random size and location) is cropped. Crops are then resized to be 224×224. A paired image is then created in the same way, except that the random square crop is shifted one pixel diagonally. So they have 1000 pairs of images. Each pair of images are highly similar as the difference is only a shift of 1 pixel. Difference is almost imperceptible for human eyes and they should be classified in the same way by a CNN.

They tested 3 state-of-the-art CNNs: VGG16, InceptionResNetV2 and ResNet50. The Top-1 prediction was different for 5% of the 1000 pairs for InceptionResNetV2 and VGG16 and even 15% for ResNet50. So this one pixel shift would introduce performance losses up to 15%.

They suggest two possible improvements, both as partial solutions:
— antialiasing of intermediate representations in CNN;
— increasing data augmentation.

To summarize, the authors of [AW19] express some concerns about the generalization power of CNNs against pre-processing for image recognition. For one particular pre-processing

Table 4.2 – Testing accuracy (in %) for GMM- and CNN-based methods. Full/full means that both training and testing sets are patches from full-sized images. Mix/mix means that both training and testing sets have been pre-resized. Resizing factors are selected randomly from $[0.48, 1.27]$.

	GF	MF	USM	WGN	JPEG	AVG
GMM full/full	91	86	97	98	89	92
GMM on mix/mix	65	85	89	86	80	81
CNN full/full	79	85	91	86	79	84
CNN on mix/mix	65	74	84	89	74	77

(not the one mentioned earlier in this subsection, a more complex one), they show that it could cause up to 30% chance that top prediction from CNN is changed (*i.e.* up to 30% loss of accuracy for the Top-1 prediction). They claim that data augmentation (*i.e.* training with more data and more diverse pre-processing pipeline) is only a partial solution and does not provide a full generalization power.

We draw some similar conclusions regarding data augmentation. We have trained detectors, one based on GMM and another based on CNN (see Section 4.3.2 and Section 4.3.1 for details on detectors) either on a dataset of full-sized images or a dataset composed of samples with pre-resizing of various factors. The testing set is consistent with the training set. Training with a mix of pre-resizing factors is data-augmentation and it should induce a good generalization power against pre-resized test samples. In Table 4.2, the results are lower for the scenario with pre-resizing of train and test, indicating that the problem is harder and data augmentation can only partially mitigate it.

4.2 Statistical tests

Decreases in performances seem to indicate a difference between the source and target distributions. It is also intuitively quite clear that pre-resizing produces changes in local statistics of the image. Therefore, the objective of this section is to exhibit theoretical justifications for these intuitions.

The GMM trained on source data exhibits lower log-likelihood on target data. However, GMM is a parametric model and this could only indicate that the problem is the parametrization and not the data. Non-parametric test is suited to investigate differences between source and target. The work from Gretton *et al.* [Gre+07] on the two-sample problem with Maxi-

mum Mean Discrepancy (MMD) is a good tool for this purpose, as this method is robust to high dimensions thanks to the use of a kernel.

4.2.1 MMD distance

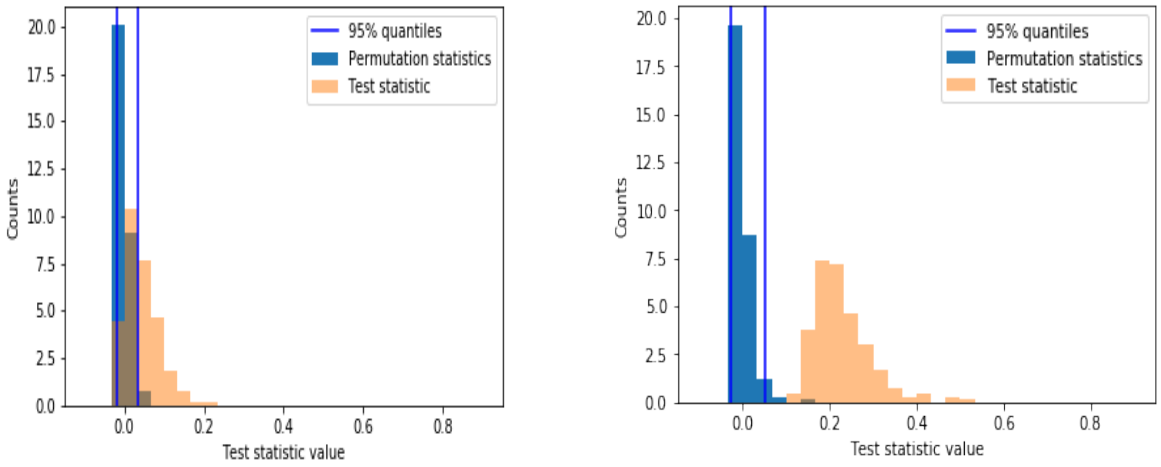
"MMD" distance is defined as:

$$MMD[\mathcal{F}, p, q] = \sup_{f \in \mathcal{F}} (\mathbb{E}_{x \sim p}[f(x)] - \mathbb{E}_{y \sim q}[f(y)]), \tag{4.1}$$

where p and q are the distributions of respectively the source and the target, x and y are source and target samples, $\mathcal{F}$ is a unit ball in a universal Reproducing Kernel Hilbert Space (RKHS) (in our case we choose Gaussian kernel with $\sigma = 2$) and $\mathbb{E}$ is the expectation operator. If $p = q$ then $MMD = 0$. This approach aims at finding the biggest difference between any moments of the two distributions: if it is zero then the distributions are equal.

In a finite sample setting, it is only possible to compute an estimation of the real MMD. It means that even if $p = q$, MMD would never be exactly 0 due to the variance of the MMD estimation (though the estimator is unbiased). It would only asymptotically converge to 0 as the number of samples grows. A permutation (randomization) test is a way to cope with this finite setting limitation. The idea is to compute the MMD on pairs (15000 in our case) of random samples composed of a mix of source and target data multiple times (200 in our case). This estimates an empirical distribution of the MMD as if $p = q$, as samples are mixed between train and target. In practice, 1.5×10^6 samples are used and not $200 \times (15000 \times 2) = 6 \times 10^6$, because only pairs of samples matter. For instance, with 5 samples it is possible to construct $\binom{2}{5} = 10$ distinct pairs of samples. So, 15000 random pairs are drawn without replacement 200 times among the 1.5×10^6 samples. For the second part of the test, the same procedure is reproduced with pairs of samples coming from source and target separately (not mixed like previously). In Equation 4.1, it means that x is drawn from the source and y from the target. The computation is also performed 200 times, on pairs of 15000 samples. Usually with a permutation test, only one estimation is computed in the second part of the test with all samples. This value is then compared with the histogram obtained in the first part of the test.

Here, 200 computations are performed on random samples to be able to compare two histograms of MMD instead of setting an arbitrary (like the usual 95%) threshold and conclude only on one particular realization of the $MMD[\mathcal{F}, p, q]$ as we consider it is a weak argument. Therefore, we propose to compare two histograms: MMD with (as if $p = q$) and without (empirical distance between p and q) permutation.



(a) (b)

Figure 4.3 – Histograms of MMD distances between training and testing data (both sets contain half of sharpened patches): (a) on original-sized testing images, and (b) on testing images subject to resizing pre-processing (bi-cubic interpolation with a factor of 0.53).

4.2.2 Distance between source and target

We have implemented a permutation test with MMD distance to test the H_0 hypothesis (train and test share the same distribution), when test is pre-resized and train is not pre-resized. Results are presented in Figure 4.3. On original-sized images (Figure 4.3.(a)), the permutation and test statistics about MMD follow roughly the same empirical histogram (same shape). So H_0 cannot be rejected. On pre-resized images, these statistics are different between permutation and test statistics (Figure 4.2.(b)). It means that MDD distances are quite different between permuted samples and samples coming from train and test one by one. Therefore, here H_0 can be rejected. This test indicates that pre-processing does change the original distribution of the images. This is quite intuitive, nevertheless here we provide theoretical evidence for it. In the following, we also present an argument in favor of statistical dependence between source and target which motivates our work.

4.2.3 Dependence between source and target

We have considered Hilbert-Schmidt independence criterion (HSIC) test. This test is described in details in [Gre+08]. HSIC test is basically an MMD two-sample test between the joint law of source and target ($P_{X,Y}$), and the product of the marginals ($P_X * P_Y$). The H_0

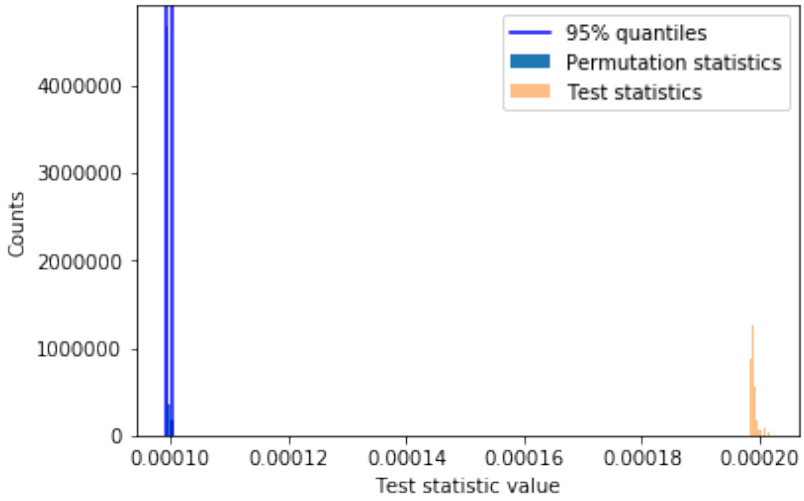


Figure 4.4 – HSIC test for the case with sharpening as manipulation and pre-resizing factor of 0.53.

hypothesis is: $P_{X,Y} = P_X * P_Y$, which would indicate that X and Y are independent. The results are shown in Figure 4.4. The test statistic values are clearly outside the permuted statistics values, so we can reject here the null hypothesis. Discarding H_0 is a hint in favor of dependence between source and target. It was an indicator for us that it is indeed possible to adapt knowledge of the source on the target. Again, this dependence is quite intuitive and it is now more formally characterized.

To summarize, the first test has given us some non-parametric arguments in favor of differences in statistical distributions of original-sized images (source) and pre-resized images (target) that could explain the reported performance drops. The dependency between the two (as reflected by the second test) motivates our domain adaptation approaches.

4.3 Classification pipelines

We describe here in details the two classification pipelines that are investigated for adaptation in the latter paragraphs.

4.3.1 Bayar and Stamm's CNN

To the best of our knowledge, no existing CNN-based method considers or reports results on small patches of 8×8 pixels. Nevertheless, for comparisons purposes, we improve and

adapt the state-of-the-art deep-learning-based method from Bayar and Stamm [BS16], so that the CNN can work with very small patches. Indeed, with the four pooling layers in the original CNN of [BS16] and 8×8 patches as input, the output size of these pooling layers drops to 1×1 and the following 2D convolution cannot be computed anymore. It is technically possible to keep one or two of the four pooling layers to work with 8×8 patches, but experimentally we have obtained better results without any pooling layer retained. We think that pooling as spatial reduction may have a good effect for big patches (for example 256×256 pixels), but not anymore on 8×8 patches where we should probably avoid information loss caused by pooling. The learning rate has also been tuned for better accuracy from 10^{-3} (the value suggested in the original paper [BS16]) to 10^{-4}. We use the Caffe implementation from the authors (https://gitlab.com/MISLgit/constrained-conv-TIFS2018) on exactly the same training and testing patches as GMM (with the same number of them as well).

4.3.2 Gaussian Mixture Model

The classifier presented here is largely inspired by the method of Fan *et al.* [FWC15] as it is one of the state-of-the-art methods for detecting manipulations on small patches. Firstly, Gaussian Mixture Models (GMMs) are trained, one for each set of patches (original, Gaussian filtered, median filtered, *etc.*), leading to six models in total (see Table 4.1). Models are trained to maximize the likelihood on patches with the Expectation-Maximization (EM) algorithm. The log-likelihood for sample $\boldsymbol{x}_l$ under a mixture of N Gaussian components, parameterized by $\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{,k=1,2,\ldots,N}$, is:

$$\mathcal{L}(\boldsymbol{x}_l|\boldsymbol{\theta}) = \log\left(\sum_{k=1}^{N} \pi_k \mathcal{N}(\boldsymbol{x}_l|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right), \tag{4.2}$$

where π_k, $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ are respectively the weight, mean and multivariate (full) covariance matrix for the k^{th} component in the mixture. Here, the DC component of each patch is removed so the patch mean is 0 thus $\boldsymbol{\mu}_k$ are all zeros. After the GMMs are trained, a very quick and efficient technique to produce a decision for a testing sample is to compute the log-likelihood for each GMM and compare these values. In the case of binary classification, it means calculating the log-likelihood ratio between the GMMs of manipulated patches and that of the original patches [FWC15], as:

$$r(\boldsymbol{x}_l) = \frac{\mathcal{L}_{GMM_{manip}}(\boldsymbol{x}_l)}{\mathcal{L}_{GMM_{ori}}(\boldsymbol{x}_l)}. \tag{4.3}$$

If the ratio $r(\boldsymbol{x}_l) > 1$, then the decision should be that the sample patch $\boldsymbol{x}_l$ is a manipulated one, otherwise it is original. This configuration is used as a classifier to adapt for the

fully unsupervised scenario described in Section 4.4.4. For the weakly-supervised scenario (Section 4.4.3, we add another stage in the pipeline as explained in the following.

In the first step of the EM algorithm (E step), we need to compute component scores which are likelihood values with respect to each Gaussian component in the GMM:

$$r_k^{(l)} = \pi_k \mathcal{N}(\boldsymbol{x}_l | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k). \tag{4.4}$$

We notice that these component scores form a more detailed descriptor than the log-likelihood value for patch $\boldsymbol{x}_l$. Therefore, we propose to use them as features to feed a classifier. For binary classification, the feature vector $(r_{1,ori}^{(l)}, r_{2,ori}^{(l)}, \ldots, r_{N,ori}^{(l)}, r_{1,manip}^{(l)}, r_{2,manip}^{(l)}, \ldots, r_{N,manip}^{(l)})$ of dimension $2N$ for the patch sample $\boldsymbol{x}_l$ is a concatenation of component scores of the two trained GMMs under comparison, each having N components. We use a small Dense Neural Network (DNN) as a classifier. It is a very simple network with two hidden layers of respectively 256 and 128 neurons, ReLU activation, dropout of 0.5 and Adam optimizer with default parameters for minimizing the *cross-entropy* loss. This architecture has not been optimized as it is not a crucial part given that the classification is quite easy.

As expected, experiments show that performances of the baseline scenario (*i.e.*, when testing samples are not pre-resized) are almost identical to the detector based on the log-likelihood ratio. In Section 4.4.3, we propose a weakly-supervised adaptation method for this classification pipeline composed of GMMs and DNN.

1) Practical implementation

The code is available online [1] to reproduce experiments. We use 8×8 patches. They are flattened (vectorized) and centered (the mean of each patch is removed), as in [FWC15]. We use the Scikit-learn [Ped+11] implementation of Gaussian Mixture Model. Each GMM has 75 components. This number is a good trade-off between model complexity for the training phase and performance on the testing phase. In order to counterbalance the weaknesses of the EM algorithm, we perform initialization five times for mixture weights π_k and covariance matrices Σ_k. The initialization is done using K-means. The GMM means μ_k are initialized to zeros and forced again to be zero after training [ZW11]; [FWC15]. More details about the practical implementation can be found in [DWC20]; [DWC19b].

2) Base score of the GMM pipeline

We provide in Table 4.3 the baseline scores for the GMM pipeline. It summarizes the performance loss for individual manipulation as well as the average for all manipulations. The average drop of performance is -17.5%. The side-effect of a resizing ratio of 0.5 would be that pixels are downsampled and not interpolated. Therefore we avoid using such ratios.

1. https://forge.uvolante.org/darmet/GRAFT

Table 4.3 – Testing accuracy (in %) without any adaptation for GMM-based method using log-likelihood ratio (details in Section 4.3.2). The first column gives the pre-resizing factors. The performance drops compared to the case without pre-resizing (*i.e.*, the row of $\times 1$) is given in parentheses. We do not use ratios like 0.5 to avoid the potential special side-effect of such ratios. The last column of "AVG" gives the average accuracy and performance drop of the 5 classification problems.

	GF	MF	USM	WGN	JPEG	AVG
$\times 1$	91	86	97	98	89	92
$\times 0.51$	64 (-27)	75 (-11)	73 (-24)	69 (-29)	79 (-10)	72 (-20)
$\times 0.76$	78 (-13)	81 (-5)	81 (-16)	73 (-25)	84 (-5)	79 (-13)
$\times 1.15$	55 (-36)	80 (-6)	87 (-10)	85 (-13)	79 (-10)	77 (-15)
$\times 1.25$	51 (-40)	75 (-11)	74 (-23)	81 (-17)	67 (-22)	70 (-22)

Table 4.4 – Testing accuracy (in %) for the GMM-based method by using less training patches.

	GF	MF	USM	WGN	JPEG	AVG
Base score with 150000 patches	81	83	95	90	84	87
Base score with 400000 patches	82	84	96	91	87	88

Table 4.5 – Testing accuracy (in %) for the GMM-based method. Both training and testing have been conducted on patches of pre-resized images with a specific factor. The GMMs training uses 800000 patches, the same as for the training of GMMs on original-sized images.

	GF	MF	USM	WGN	JPEG	AVG
Resizing $\times 0.51$	81	87	92	95	83	88
Resizing $\times 1.25$	75	84	98	94	85	87

3) Additional results of GMM training

We have investigated potential limitations of the GMM pipeline, *i.e.* when less samples are available for training. We first report in Table 4.4 some results of performances of GMMs when trained on original-sized images with less samples, respectively 150000 and 400000 patches (800000 samples are used for experiments in the manuscript). In general, the classification accuracy increases when more patches are used for GMM training. In order to have a satisfying average base score (*i.e.*, an average accuracy higher than 90%), we have decided to use 800000 patches to train GMMs as in the thesis manuscript.

Then we provide results for GMMs trained on 800000 patches extracted from pre-resized images of a specific scaling factor (see Table 4.5). Intuitively, these results can be considered as the "best" results that we can achieve on patches of pre-resized images. We will see that

the average accuracy of our proposed unsupervised adaptation method named GRAFT (in Table 4.9, Table 4.10 and Table 4.11) is quite close to the result of training from scratch on 800000 patches from pre-resized images, which demonstrates the effectiveness of GRAFT. Detailed analysis will be presented in Section 4.4.4.2.

4.4 Adaptation

As depicted in introduction of Section 4.1, to deal with these performance losses induced by statistical discrepancies between train and test, it is possible to adapt either the features or the model. Our work is focused on model adaptation, and especially GMMs, though we present in Section 4.4.1 briefly two feature-based approaches (*i.e.* optimal transport and kernel methods) and references. Then we introduce briefly in Section 4.4.2 fine-tuning of CNN which can be considered as a kind of model adaptation, and more precisely adaptation of the CNN designed by Bayar ans Stamm [BS16] with some experimental results. Finally, we present two adaptation methods for GMM in details. First, one relies on few labels on the target, *i.e. weakly-supervised* (Section 4.4.3). The other technique is fully unsupervised and we name it GRAFT for GMM Resizing Adaptation by Fine-Tuning (Section 4.4.4). Some results of the two methods are presented. Details of the classification pipeline of the original GMM method, without adaptation, can be found in Section 4.3.2. A last section (Section 4.4.5) compares the three model adaptation techniques.

4.4.1 Feature adaptation

The idea of feature adaptation is to find a transformation for source features that would allow a classifier learned on transformed source features to perform better on target features than a classifier learned on the original source features. In the case of Optimal Transport (OT), the source is usually moved towards the target. For kernel methods, both source and target are transformed to be closer. In both cases a classifier should be re-trained on transformed source features. For model adaptation techniques, the classifier is learned directly on the original source data and it is later adapted to the target without retraining.

OT-based adaptation [Cou+17] tries to preserve conditional distributions: $P_{train}(y|x_{train}) = P_{test}(y|T(x_{test}))$ with y representing the labels and T a transport map for test samples. It aims at finding a probabilistic coupling γ between domains Ω_{train} and Ω_{test} supplied with two probability measures μ_{train} and μ_{test} and a cost function c that allows computing the cost of moving a point from Ω_{train} to Ω_{test}. This probabilistic coupling γ provides a bijective

transport map T. The cost function is in our case the Euclidean distance between samples in the feature space. The rigorous formulation, initially from Kantorovich (1942), is [Cut13]:

$$\begin{aligned} \gamma_0 &= arg\min_{\gamma} \int_{\Omega_{train}\times\Omega_{test}} c(x,y)\gamma(x,y)dxdy, \\ \text{s.t.} \quad &\int_{\Omega_{train}} \gamma(x,y)dx = \mu_{train}, \int_{\Omega_{test}} \gamma(x,y)dy = \mu_{test}, \end{aligned} \tag{4.5}$$

which leads to an optimization problem that can be solved through linear programming. The optimal value obtained with γ_0 is called Wasserstein distance, which defines a distance between two probability measures. After finding the optimal coupling γ_0, the training samples are transported on the testing ones and a classifier is trained with these transported samples and their labels.

Another approach is based on kernel methods [Pan+11]. Basically the idea is to minimize:

$$Dist(X'_{train}, X'_{test}) = \left\| \frac{1}{n_{train}} \sum_{i=1}^{n_{train}} \Phi(x_{i,train}) - \frac{1}{n_{test}} \sum_{j=1}^{n_{test}} \Phi(x_{j,test}) \right\|_{\mathcal{H}}^2, \tag{4.6}$$

where $X'_{train} = \{\Phi(x_{i,train})\}$, $X'_{test} = \{\Phi(x_{j,test})\}$, and Φ is a parametric kernel. As explained in [Pan+11], solving this problem is equivalent to computing the eigenvectors of matrices based on kernel distance between samples in feature space which is quite straightforward and easy to implement.

SPAM [PBF10] and SRM[FK12] features are state-of-the-art features for image forensics and performance losses are also observed. Therefore, we have tried both approaches to counterbalance effect of pre-resizing but results are not satisfactory.

The performances are downgraded by adaptation when transporting the source towards the target and re-training a classifier on transported source samples (standard framework for domain adaptation through optimal transport). The performances remain almost the same when transporting the target towards the source without retraining. Our intuition is that the feature spaces are too high-dimensional and complex (the dimension is 1250 for SPAM features with its parameter $T = 3$). The OT approach requires the computation of distances between samples (cost function c). In such a high-dimensional space, it is very likely that the curse of dimensionality occurs and thus the cost function c is not reliable. In the original paper [Cou+17], the authors adapt a classifier from one manuscript digit dataset to another. They use SIFT to transform each image into a 800-bin histogram and then the histograms are normalized and the feature is only a 1-dimensional standard score. Kernel-based methods also rely on pairwise distances. These distances are computed thanks to a kernel, so the effects of dimensionality should be mitigated. We do not observe this in practice however. Beside

these considerations on dimensionality, and based on the fact that results are better for the OT-based approach when the classifier is not re-trained, we suspect transformed features have lower discriminative capabilities. That could also explain the bad results of the kernel-based method.

This problem is discussed theoretically by Ben-David *et al.* [BD+06]. They provide a generalization bound error for domain adaptation that depends on both the distance between the distributions of source and target (brought closer by OT or kernel method here), and the performances on the source (*i.e.* how good are the features for the problem). The difficulty is that these two terms are not independent. By reducing the distance between source and target, performances on the source could be damaged and *vice versa*. The theoretical results of [BD+06] give a plausible explanation for the difficulty of feature adaptation in our scenario. More formally, OT-based and kernel-based approaches reduce the gap between the two distributions (source and target) which at the same time decreases the performance on the source. Nevertheless, we believe that a feature-based method has a great potential and we consider it as a promising future research line.

4.4.2 Neural network fine-tuning

The best known technique for model adaptation is the fine tuning of a convolutional neural network pre-trained on ImageNet [Yos+14]. Classical CNNs need millions of images to train on, as in ImageNet, and a considerable amount of computing time and resources. For other problems, these conditions may not be satisfied. One possible shortcut is to use pre-trained weights on ImageNet as an initialization and then only fine-tune the last few layers. While this technique is effective, it needs access to labels in the target domain to be applied. The fine-tuning of a dense neural network is used for image manipulation detection in Section 4.4.3 to adapt to pre-resizing in a weakly-supervised setting.

The state-of-the-art results for image manipulation detection with a CNN were obtained by the CNN of Bayar and Stam [BS16]. Therefore, we use this CNN classifier to illustrate the effectiveness of CNN fine-tuning. However, these authors have designed their network to work with patches of size at least 64×64 pixels. In order to make the CNN work with 8×8 patches, one or some of the four *pooling* layers of the original network have to be removed. Otherwise the output size of these layers drop to 1×1 and the following 2D convolution is not possible anymore. We have tried different configurations and numbers of retained pooling layers (0, 1 or 2 retained layers are technically possible) and found that we obtain better performance without any pooling. This is understandable as pooling layers would cause loss of information. Performances are also better with a learning rate of 10^{-4} instead of 10^{-3}

Table 4.6 – Testing accuracy (in %) of an improved version of Bayar and Stamm's CNN-based method [BS16], with and without fine-tuning. The improved accuracy of weakly-supervised fine-tuning, compared to the setting without fine-tuning, is given in parentheses. The baseline testing accuracy without any resizing pre-processing is given in the second row.

	GF	MF	USM	WGN	JPEG	AVG
Without resizing	79	85	91	86	79	84
Resizing $\times 0.51$ (without fine-tuning)	68	82	80	66	76	74
Resizing $\times 0.51$ (fine-tuning)	73 (+5)	82 (+0)	85 (+5)	69 (+3)	77 (+1)	77 (+3)
Resizing $\times 0.76$ (without fine-tuning)	73	83	89	80	79	81
Resizing $\times 0.76$ (fine-tuning)	74 (+1)	83 (+0)	89 (+0)	80 (+0)	79 (+0)	81 (+0)
Resizing $\times 1.15$ (without fine-tuning)	59	76	79	86	61	72
Resizing $\times 1.15$ (fine-tuning)	67 (+8)	80 (+4)	90 (+11)	86 (+0)	66 (+5)	78 (+6)
Resizing $\times 1.25$ (without fine-tuning)	55	72	74	82	55	68
Resizing $\times 1.25$ (fine-tuning)	65 (+10)	77 (+5)	91 (+15)	86 (+4)	60 (+5)	76 (+8)

(the value suggested in the original paper [BS16]). Results reported in Table 4.6 have been obtained with these improved settings, the best that we could get after many experiments. We used the Caffe implementation from the authors available at https://gitlab.com/MISLgit/constrained-conv-TIFS2018.

The results for Bayar and Stamm's CNN [BS16] on pre-resized and un-resized (*i.e.* resizing factor of 1) testing sets are presented in Table 4.6. Without fine-tuning, drops in performance are severe, for instance with an upscaling factor of $\times 1.25$, the classifier is 24% less powerful to detect Gaussian Filtering and -16% on average. The average performance drop without fine-tuning for the presented resizing factors in Table 4.6 is about -13%. Drops are less severe when the resizing factor is close to 1.

We have tried different fine-tuning strategies to find the best one, *i.e.*, fine-tuning the first few, the last few, or all layers of the network. We found that only fine-tuning the dense layers at the end of CNN gives slightly better performance than the other strategies. The

learning rate has been reduced to 10^{-5}. This particular setting (adapted learning rate and fine-tuning) gave us the best adaptation performance in our experiments. Fine-tuning of the CNN is performed on the same 2000 pre-resized patches per class as in the following work Section 4.4.3 in order to be able to compare both approaches. Adaptation of CNN helps to improve the accuracy under resizing pre-processing, especially for upsampling, as shown in Table 4.6.

4.4.3 Weight adaptation and fine-tuning for Gaussian Mixture Models

As described in Section 4.3.2, we use here a classification pipeline based on GMMs for feature extraction and a small DNN for classification. Two GMMs are trained, one on original patches and another one on manipulated patches. The sample's vectors of responsibilities from the two GMMs are then concatenated and used as features. The DNN performs a classification with these features as inputs. The proposed weakly-supervised adaptation method comprises two sub-steps. First, GMMs are adapted so that they better fit to the testing samples which have undergone the resizing pre-processing. Then the DNN classifier is adapted by fine-tuning the network. Both steps are accomplished in a weakly-supervised manner, *i.e.* by using a very limited number of labeled testing samples of 8×8 patches from pre-resized images.

4.4.3.1 Method

In the following, we first show that if DC components of patches are removed (*i.e.*, Gaussian component's means $\boldsymbol{\mu}_k = \mathbf{0},\ \forall k = 1, 2, \ldots, N$), the weighted sum of covariance matrices of a GMM is equal to the covariance matrix of the data. We have $\boldsymbol{X} = (X_1, X_2, \ldots, X_p)$ a multi-dimensional random variable and $p = 64$ because patches are 8×8. Let f be the *Probability Density Function* (PDF) of the random variable $\boldsymbol{X}$ with the DC component removed and let f_k be the PDFs of each component of the related Gaussian mixture, then we have:

$$f(\boldsymbol{x}_l) = \sum_{k=1}^{N} \pi_k \ f_k(\boldsymbol{x}_l) = \sum_{i=1}^{N} \pi_k \ \mathcal{N}(\boldsymbol{x}_l | \mathbf{0}, \boldsymbol{\Sigma}_k), \tag{4.7}$$

where $\boldsymbol{x}_l$ is a p-dimensional sample of the random variable $\boldsymbol{X}$, $\boldsymbol{\Sigma}_k$ and π_k are respectively the covariance and the weight for the k^{th} component in the mixture. Now let us compute the elements of the covariance matrix of $\boldsymbol{X}$:

$$\begin{aligned} &\text{cov}(X_i, X_j) = \mathbb{E}_f[X_i X_j] - \mathbb{E}_f[X_i]\,\mathbb{E}_f[X_j] = \mathbb{E}_f[X_i X_j] \\ &= \sum_{k=1}^{N} \pi_k \,\mathbb{E}_{f_k}[X_i X_j] = \sum_{k=1}^{N} \pi_k (\boldsymbol{\Sigma}_k^{(i,j)} + \boldsymbol{\mu}_k^{(i)} \boldsymbol{\mu}_k^{(j)}) = \sum_{k=1}^{N} \pi_k \boldsymbol{\Sigma}_k^{(i,j)}, \end{aligned} \tag{4.8}$$

where superscripts $(i), (j)$ and (i, j) are element indices within the corresponding vector and matrix. Considering that variance is a special case of covariance, from Equation 4.8 we can see that the covariance matrix of the data is equal to the weighted sum of the covariance matrices of the Gaussian mixture.

We assume that we have only a few labeled samples on the target domain, not enough to train a model from scratch (this would need around 200000 samples of each class) but enough to compute an empirical covariance matrix per each class on the target (around 1000 samples for each class). The GMMs' parameters should be slightly adjusted so as to enhance the descriptive capability of the model on the target data. GMMs can be adjusted in two ways: adjusting the weights or the covariance matrices (the means are zeros). Beside that, our aim is to have a quick adaptation solution. Therefore, we choose to adapt the GMM weights. The weights contain less parameters (only a vector and not matrices). Adaptation of GMMs' weights can be formulated as an optimization problem:

$$\begin{aligned} &\underset{w_k}{\text{minimize}} \left\| \left(\sum_{k=1}^{N} w_k \times \mathbf{\Sigma}_k \right) - \mathbf{\Sigma}_{data} \right\|_F \\ &\text{subject to } \sum_{k=1}^{N} w_k = 1, \text{ and } \ 0 < w_k < 1, \ \forall k = 1, 2, \ldots, N. \end{aligned} \tag{4.9}$$

In Equation 4.9, the w_k's are adapted GMM weights to be deduced, $\mathbf{\Sigma}_{data}$ is the empirical covariance matrix on the target domain, and F stands for the Frobenius norm. We do acknowledge that semi-definite positive matrices lie on a Riemannian manifold, thus with a curvature. So a geodesic distance would be more adapted. However, we did not notice any differences in classification performances or in the results of optimization using the Euclidean distance instead of the geodesic distance. Although the geodesic distance is more expensive and slower to compute. Therefore, in practice we use the Frobenius norm.

These adjustments of weights for each GMM can be seen as a fine-tuning for feature extraction. It is a means of reducing discrepancy between features of source and target domains. In the second step of our method, *classifier adaptation* by fine-tuning the DNN is carried out to cope with drifts in features and therefore enhance the discriminative capability of the classifier.

4.4.3.2 Results

1) With DNN fine-tuning only

DNN fine-tuning helps to improve the detection accuracy, but sometimes the improvement

Table 4.7 – Testing accuracy (in %) of DNN fine-tuning, combined without or with weights adaptation of GMMs. The improved accuracy of weakly-supervised adaptations, compared to the setting without adaptation, is given in parentheses. Testing accuracy without resizing is also given in the second row for reference. The last column of "AVG" presents the average accuracy (and average accuracy improvement in parentheses, if any) of the 5 classification problems.

	GF	MF	USM	WGN	JPEG	AVG
Without resizing	91	86	97	98	89	92
Resizing $\times 0.51$ (without adaptation)	64	75	73	69	79	72
Resizing $\times 0.51$ (DNN fine-tuning only)	72 (+8)	75 (+0)	91 (+18)	71 (+2)	82 (+3)	78 (+6)
Resizing $\times 0.51$ (GMM adaptation + DNN fine-tuning)	78 (+14)	76 (+1)	92 (+19)	70 (+1)	86 (+7)	80 (+8)
Resizing $\times 0.76$ (without adaptation)	78	81	81	73	84	79
Resizing $\times 0.76$ (DNN fine-tuning only)	78 (+0)	77 (-4)	92 (+11)	81 (+8)	84 (+0)	82 (+3)
Resizing $\times 0.76$ (GMM adaptation + DNN fine-tuning)	83 (+5)	82 (+1)	94 (+13)	85 (+12)	84 (+0)	86 (+7)
Resizing $\times 1.15$ (without adaptation)	55	80	87	85	79	77
Resizing $\times 1.15$ (DNN fine-tuning only)	66 (+11)	82 (+2)	95 (+8)	96 (+11)	79 (+0)	84 (+7)
Resizing $\times 1.15$ (GMM adaptation + DNN fine-tuning)	70 (+15)	85 (+5)	95 (+8)	96 (+11)	82 (+3)	86 (+9)
Resizing $\times 1.25$ (without adaptation)	51	75	74	81	67	70
Resizing $\times 1.25$ (DNN fine-tuning only)	63 (+12)	78 (+3)	95 (+21)	90 (+9)	70 (+3)	79 (+9)
Resizing $\times 1.25$ (GMM adaptation + DNN fine-tuning)	66 (+15)	80 (+5)	95 (+21)	95 (+14)	78 (+11)	83 (+13)

is rather limited (see Table 4.7, rows of "DNN fine-tuning only"). By fine-tuning, the classifier's decision boundary is slightly adjusted, somehow similar to the case of selecting a new

threshold for the comparison of likelihood (instead of 1 initially). In order to further enhance the discriminative power of the whole forensic pipeline, it is necessary to also adapt GMMs, the underlying feature extractor, which are until now trained solely on the source data while being "blind" to the target domain. Therefore, GMMs should be tweaked, more precisely their weights, in order to better fit the target data.

2) With GMM weights adaptation

We observe in Table 4.7 some clear improvements (*e.g.*, for WGN and JPEG under upsampling of $\times 1.25$) when fine-tuning of DNN is conducted jointly with GMM weights adaptation. In addition, there is consistent average accuracy improvement under all the considered resizing factors (see last column of Table 4.7) for adaptation of both GMMs and DNN, when compared to DNN fine-tuning only. The standard deviation of the results is under 10^{-1}. The acccuracy increase offered by adaptation of GMMs and DNN depends on manipulations. For example, our method is able to recover up to $+19\%$ for sharpening (USM) and resizing of $\times 0.51$, but there are not such improvements for median filtering (MF). Median filtering is the manipulation with the smallest score (86% in Table 4.3, row of $\times 1$) on baseline (without resizing of the testing set) and is also across resizing factors one of the hardest to deal with. Beside that, our method works better with upsampling (for example $+13\%$ for the average accuracy improvement with a factor of $\times 1.25$ and less for factors $\times 0.51$ and $\times 0.76$). Our conjecture is that with downsampling, some striking local dependencies in patches are partially removed so it needs more complex transformation than a simple weights adjustment to allow a GMM to describe them well. With upsampling, the dependencies are somehow mildly smoothed so it is easier to adapt.

3) Mixed resizing factors

Our method also performs well with a mix of resizing factors. As shown in Table 4.8, our method still obtains good results when factors are randomly drawn within an interval (following the uniform distribution). This is not a surprise since our method only intends to adapt the GMM-based feature extractor to the new covariance of the data and the DNN classifier to these new features, without taking into account the specific factor value and algorithm of the resizing pre-processing.

4.4.4 GRAFT: GMM Resizing Adaptation by Fine-Tuning

Quite similarly with previous Section 4.4.3, our objective here is to come up with a simple method to quickly adapt a GMM to the target dataset (*i.e.* pre-resized), starting from a model learned on the source dataset (*i.e.* without pre-resizing). However, we aim here for an unsupervised method, while the method of previous Section 4.4.3 follows a weakly-

Table 4.8 – Testing accuracy (in %) of DNN fine-tuning, combined without or with weights adaptation of GMMs, for the case of mixed resizing factors. The resizing factor is drawn following the uniform law within the specified interval.

	GF	MF	USM	WGN	JPEG	AVG
Resizing $\times[0.48, 0.72]$ (without adaptation)	71	81	76	72	87	77
Resizing $\times[0.48, 0.72]$ (DNN fine-tuning only)	78 (+7)	81 (+0)	91 (+15)	76 (+4)	87 (+0)	83 (+6)
Resizing $\times[0.48, 0.72]$ (GMM adaptation + DNN fine-tuning)	83 (+12)	80 (-1)	92 (+16)	80 (+8)	88 (+1)	85 (+8)
Resizing $\times[1.12, 1.27]$ (without adaptation)	53	78	81	83	74	74
Resizing $\times[1.12, 1.27]$ (DNN fine-tuning only)	63 (+10)	78 (+0)	95 (+14)	91 (+8)	75 (+1)	80 (+6)
Resizing $\times[1.12, 1.27]$ (GMM adaptation + DNN fine-tuning)	64 (+11)	83 (+5)	98 (+17)	95 (+12)	78 (+4)	84 (+10)

supervised framework. We remind that as patches are centered, free parameters of the GMMs are the weights and the covariance matrices. In previous Section 4.4.3, the weights of the GMM are adapted. Here, covariance matrices are modified and weights are left untouched. This is motivated by the analysis of Fan *et al.* in [FWC15] on the leading eigenvectors of GMM covariance matrices. The authors showed that the leading eigenvectors of GMM covariance matrices are indeed very different for the model trained on original patches and the one trained on manipulated patches. This seems to indicate that the discriminative power of the method [FWC15] is closely related to the learned GMM covariance matrices. This motivates us to find an adaptation procedure for GMM covariance matrices.

We focus on the adaptation of covariance matrices in GMMs. Therefore, we briefly present here previous works on transformation of covariance matrices. In [Zan+18]; [RJC19], the authors use covariance matrices between signals of several electrodes placed on the head of the subject as features. These features are used to distinguish movements made by the subject. Their objective is to adapt features between experiments to avoid re-calibration, in a semi-supervised setting. Domain adaptation is performed through covariance matrix transformations. More precisely, in [Zan+18]; [RJC19] they compute centers of mass with geodesic distance (*i.e.* the geometric mean in the language of Riemannian geometry) for the training and the testing ensembles of covariance matrices and minimize the geodesic distance

between the two. It is worth pointing out that for us, the covariance matrices are part of the classification model and not the features, and that we consider the challenging unsupervised setting and propose a new adaptation method as presented below.

4.4.4.1 Method

1) Outline

The inputs of the method are:

— $\mathcal{C}_1$: a set of $N = 2 \times K$ covariance matrices from two GMMs trained on the source dataset (the two GMMs, each having K components, are trained respectively on original patches and manipulated patches);

— $\mathcal{C}_2$: empirical estimations of the covariance matrices for original and manipulated patches on target pre-resized dataset[2].

We seek to obtain $\mathcal{C}_1^{adp}$, a set of N covariance matrices adapted to the pre-resized target domain. $\mathcal{C}_1^{adp}$, along with the weights of original GMMs in the source domain, constitutes two adapted GMMs which have improved forensic performance on the target domain. $\mathcal{C}_1^{adp}$ will be obtained by transforming $\mathcal{C}_1$ using information from $\mathcal{C}_2$. Of course, as an unsupervised adaptation approach, our method does not use any ground-truth label on the target domain, neither for the estimation of $\mathcal{C}_2$ nor for the transformation of $\mathcal{C}_1$.

Formally we have:

$$\begin{aligned} \mathcal{C}_1 &= \{C_1^{(1),ori}, ..., C_K^{(1),ori}, C_1^{(1),mnp}, ..., C_K^{(1),mnp}\}, \\ \mathcal{C}_2 &= \{C_1^{(2),ori}, ..., C_{\frac{M}{2}}^{(2),ori}, C_1^{(2),mnp}, ..., C_{\frac{M}{2}}^{(2),mnp}\}. \end{aligned} \tag{4.10}$$

The idea is to find some transformation of $\mathcal{C}_1$ to bring it "closer" to $\mathcal{C}_2$, under the constraint of increasing the *likelihood* of adapted GMMs on the target pre-resized dataset. Here, we have been inspired by the work of Rodrigues *et al.* [RJC19] in the brain-computer interface field, where the authors propose to use the Riemannian Procrustes Analysis (RPA) method, an adaptation between sets of covariance matrices. In their work, and unlike in ours, covariance matrices are features and not parameters of a probabilistic model. They perform feature adaptation in a semi-supervised manner while we would like to adapt our GMMs in an unsupervised scenario. The objectives are not comparable and it is not possible to directly use their method for our problem. However, our GRAFT method retains the RPA spirit by using a set of basic geometrical transformations of covariance matrices. So we briefly present here the RPA procedure which comprises three main steps (mathematical details can be found in [RJC19]):

2. In our algorithm, M estimations are computed in order to improve the robustness against the variance of empirical estimation. Details on how to obtain $\mathcal{C}_2$ are presented later in this section.

1. Translate source and target sets to obtain the identity as geometric mean (re-centering): $\mathcal{C}_1^{ctr} = T_1(\mathcal{C}_1)$ where $\mathcal{C}_1$ is the first set of covariance matrices. Let V be the geometric mean of $\mathcal{C}_1 = \{C_1^1, \ldots, C_K^1\}$. Then T_1 is defined as $C_i^{1,(ctr)} = V^{-\frac{1}{2}} C_i^1 V^{-\frac{1}{2}}$. Similarly, we have T_2 for the second set $\mathcal{C}_2$;
2. Perform rescaling on each axis to get unit variance: $\mathcal{C}_1^{str} = S_1(\mathcal{C}_1^{ctr})$ and $\mathcal{C}_2^{str} = S_2(\mathcal{C}_2^{ctr})$. S_1 is defined such that $C_i^{1,(str)} = \left(C_i^{1,(ctr)}\right)^p$ where p is a proper scaling factor for the variance normalization;
3. Minimize the geodesic distance between $\mathcal{C}_1^{str}$ and $\mathcal{C}_2^{str}$ by finding an optimal rotation U around the origin (the rotation is applied to $\mathcal{C}_1^{str}$ only, not on $\mathcal{C}_2^{str}$). The $C_i^{1,(str)}$ are modified so that $C_i^{1,(rot)} = U C_i^{1,(str)} U^T$ with $UU^T = \mathcal{I}$ (*i.e.*, the product is the identity matrix).

With the RPA procedure, both sets of features become comparable: a classifier learned with C_1 features is also meaningful with C_2 features. RPA uses geodesic distance between matrices to derive the transformation, while GMMs use likelihood to carry out classification (see Section 4.3.2 for the GMM-based classification pipeline). Hence, in GRAFT we shall use the group of elementary geometrical transformations of covariance matrices, but combine and optimize them in a different and appropriate manner.

2) Transformation and interpolation

In the first place, we transform our $\mathcal{C}_1$ and $\mathcal{C}_2$ in Equation 4.10 by using the three geometrical transformations of the RPA procedure (see above). More precisely, we first translate $\mathcal{C}_1$ and $\mathcal{C}_2$ so that they both have the identity as their barycenter. Then, stretching and rotation are performed so that $\mathcal{C}_1$ and $\mathcal{C}_2$ get closer in the transformed space in terms of geodesic distance. The set of transformed covariance matrices corresponding to $\mathcal{C}_1$ are denoted by $\mathcal{C}_1^{RPA}$. We know from the analysis at the end of last subsection that this transformation is not optimum for a GMM-based classification pipeline, mainly because RPA does not use likelihood as a criterion during the transformation. Therefore, additional steps are needed.

We then consider another transformation which translates $\mathcal{C}_1$ towards $\mathcal{C}_2$, so that the translated version, denoted by $\mathcal{C}_1^{trg}$, has the same center of mass as $\mathcal{C}_2$. With a little abuse of notation, for original patches this means satisfying the following equation (similarly for manipulated patches):

$$\sum_{i=1}^{K} \pi_i \times \mathcal{C}_i^{trg,ori} = \frac{2}{M} \sum_{j=1}^{M/2} C_j^{(2),ori}, \tag{4.11}$$

where $\mathcal{C}_i^{trg,ori}$ is the i-th covariance matrix for original patches in $\mathcal{C}_1^{trg}$, π_i are the GMM weights of original patches in the source domain, and $C_j^{(2),ori}$ is the j-th estimated covariance of original patches in the target domain. This transformation considers the fit of second-order

statistics (*i.e.*, the covariance matrix) to the target dataset. In fact, it can be shown, with the assumption of a perfect fit of GMM and centered patches, that the weighted sum of Gaussian components' covariance matrices of a GMM (the left-hand side of Equation 4.11) is equal to the covariance matrix of the data (the right-hand side of Equation 4.11 is an estimation of the data covariance).[3] Indeed, as in [ZW11], [FWC15] patch samples are centered to have zero mean (*i.e.*, the DC component is removed for each patch), and accordingly Gaussian components in GMM also each have zero mean. As proven in Section 4.4.3, the cross terms in the covariance computation disappear with zero-mean Gaussian components. Then we have:

$$\text{cov}(X_i, X_j) = \sum_{k=1}^{K} \pi_k \times \Sigma_k^{(i,j)}, \tag{4.12}$$

where X is the 64-dimensional random variable of pixel values from the vectorized patches, $\text{cov}(.,.)$ is the covariance function, Σ_k and π_k are respectively the covariance and the weight for the k-th component in GMM, and i, j and (i, j) are element indices within the corresponding vector and matrix.

Technically, similar to the translation step in RPA [RJC19] (see step 1) of RPA in the last subsection), the translation to $\mathcal{C}_1^{trg}$ is realized by simple matrix multiplications on $\mathcal{C}_1$.

Practically and theoretically, neither $\mathcal{C}_1^{RPA}$ nor $\mathcal{C}_1^{trg}$ are optimal in terms of GMM likelihood (see Figure 4.6 of the next Section 4.4.4.2 for a concrete example). The former has the identity as its geometric mean, while the latter only considers fit of covariance matrices on average but not the GMM likelihood. However, both $\mathcal{C}_1^{RPA}$ and $\mathcal{C}_1^{trg}$ get closer to the target domain to some extent, and a natural and simple idea is to *interpolate* between the two to get a better solution. This is a heuristic-based approach but it has good intuition and is experimentally effective. Another advantage is that a simple interpolation between two valid covariance matrices still leads to a valid solution as a symmetric positive semi-definite matrix.

The interpolation is naturally driven and governed by the *maximization of likelihood* of GMMs. The rationale behind is the fact that GMMs rely heavily on the descriptive capability (high likelihood) to carry out correct classification. Two distinct (regarding respectively original and manipulated classes of patches) interpolation coefficients are computed between $\mathcal{C}_1^{RPA}$ and $\mathcal{C}_1^{trg}$:

— α_1 is used to interpolate between $\mathcal{C}_1^{RPA,ori}$ and $\mathcal{C}_1^{trg,ori}$ (original patches);

— α_2 is used to interpolate between $\mathcal{C}_1^{RPA,mnp}$ and $\mathcal{C}_1^{trg,mnp}$ (manipulated patches).

Optimal values for α_1 and α_2 are computed by maximizing the log-likelihood respectively on the two GMMs. This will be shown to enhance both the descriptive power by maximizing the log-likelihood and the discriminative capability by choosing separate coefficients for

3. Detailed proof can be found in previous Section 4.4.3.

original/manipulated classes. This trade-off between adapting to the new domain while simultaneously preserving the discriminative power is also motivated by the theoretical study of [BD+06].

To estimate the covariance matrix on 8×8 patches of pre-resized testing images, the usual empirical estimator is used:

$$\sigma_{ij} = \frac{1}{N_P - 1} \sum_{l=1}^{N_P} (x_{il} - \mu_i)(x_{jl} - \mu_j), \tag{4.13}$$

where i is the row index, j is the column index, N_P is the number of patches, and μ_i, μ_j are respectively the empirical mean of row i and column j over the N_P samples. This estimator is unbiased, yet it suffers from a high variance. As explained earlier, in order to counterbalance the variance of the estimation, we perform M estimations of empirical covariance matrices on separate subsets of the target dataset of patches from pre-resized images. It is worth mentioning that we estimate matrices separately for the two original/manipulated classes, by using the so-called pseudo-labels in the target domain. Pseudo-labels are an important feature of GRAFT and are described in the next subsection.

3) Pseudo-labels

In the unsupervised framework of GRAFT, we do not have access to ground-truth labels on the target domain. However, it turns out that the accuracy of the GMM-based classifier does not drop to 50%, *i.e.* random guessing. It means that the classifier is still able to label accurately some pre-resized test samples. From these samples, we derive *pseudo-labels*. We explain in the following how these pseudo-labeled samples are selected.

At the beginning of the optimization of α_1 and α_2, their initial values are drawn randomly to compute the first two sub-optimal, interpolated GMMs. We assume that even with this sub-optimal interpolation, the GMMs can still label correctly some testing samples. Hence, we need to select almost surely original and almost surely manipulated patches. Almost surely original (*resp.* manipulated) patches are selected from the 5–15 (*resp.* 85–95) percentile of the likelihood ratio, leading to 20% patches with reliable pseudo-labels because their likelihood ratio is farthest from 1. The GMMs are originally trained on the source domain, so most surely classified target samples are very likely to be closer to the source domain than to the target domain. It means that they are not enough representative of the effect of pre-resizing. Therefore, extreme likelihood ratio values (percentile 0–5 and 95–100) are discarded.

To validate this approach, we have computed the accuracy of pseudo-labeled samples and it is typically above 95%. Of course, ground-truth labels on the target domain are not used in the GRAFT method and have been only used here to validate our hypothesis on pseudo-labels. In the end, 20% (10% almost surely original + 10% almost surely manipulated) of the test

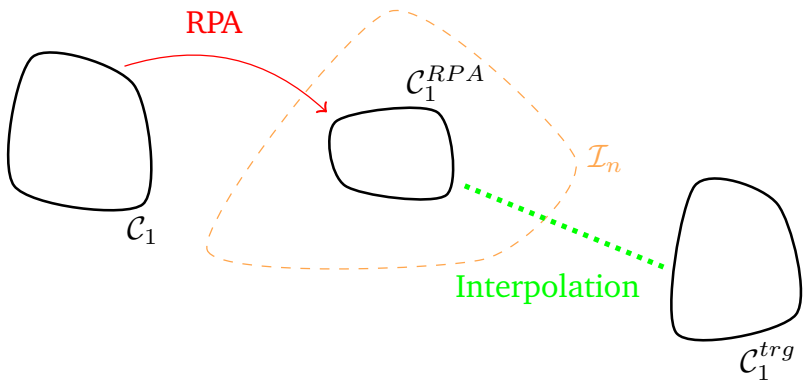


Figure 4.5 – Transformation and interpolation in GRAFT. $\mathcal{I}_n$ is the identity matrix of dimension $n \times n$, in our case 64×64, the size of the covariance matrix of vectorized 8×8 patches. The interpolation is essential in GRAFT which maximizes the log-likelihood of GMMs with regard to the target domain.

data are pseudo-labeled. The method is robust regarding the chosen percentages. Selecting percentiles 5–20 and 80–95 (or 10–15 and 85–90) would not significantly impact the final accuracy. However, the classification accuracy starts to drop when more than 30% of the test data are pseudo-labeled because their accuracy drops.

One detail is that we use directly $\mathcal{C}_1$ to get the pseudo-labels used for the estimation of empirical covariance matrices $\mathcal{C}_2$ on test samples (see steps 2 and 3 of Algorithm 1). This is different from the derivation of pseudo-labels via sub-optimal interpolation mentioned above, but remains reasonable because at the stage of estimating $\mathcal{C}_2$ it is impossible to use interpolation to get pseudo-labels. In fact, the estimation of $\mathcal{C}_2$ serves to obtain $\mathcal{C}_1^{trg}$, one end point of the interpolation.

4) Summary

The main steps of the unsupervised adaptation method of GRAFT are illustrated in Figure 4.5 and its pseudo-code is presented in Algorithm 1. The red line in Figure 4.5 represents translation toward identity, stretching and rotation. Sets of covariance matrices within the orange dashed contour have the identity matrix ($\mathcal{I}_n$) as their geometric mean. In Algorithm 1, α_1 and α_2 are the coefficients of an interpolation represented by the green dotted line in Figure 4.5. Formally, the adapted sets of covariance matrices are obtained as:

$$\begin{aligned} \mathcal{C}_{adp}^{ori} &= \mathcal{C}_1^{trg,ori} * (1-\alpha_1) + \mathcal{C}_1^{RPA,ori} * \alpha_1, \\ \mathcal{C}_{adp}^{mnp} &= \mathcal{C}_1^{trg,mnp} * (1-\alpha_2) + \mathcal{C}_1^{RPA,mnp} * \alpha_2. \end{aligned} \tag{4.14}$$

α_1 and α_2 are computed such that they maximize the sum of log-likelihood of the two GMMs:

$$\underset{\alpha_1,\alpha_2}{\operatorname{argmax}} \mathcal{LL}_{adp}^{ori}(\alpha_1) + \mathcal{LL}_{adp}^{mnp}(\alpha_2). \tag{4.15}$$

Algorithm 1 GRAFT algorithm

Input: Source and target data, two GMMs trained respectively on original and manipulated patches of source data

Output: Adapted GMMs

1: Concatenate the two sets of covariance matrices from the two trained GMMs on source data to form $\mathcal{C}_1$
2: Compute pseudo-labels based on ratio of likelihood with $\mathcal{C}_1$ as covariance matrices
3: Use these pseudo-labels to compute estimations of covariance matrices of original and manipulated patches on target data to get $\mathcal{C}_2$
4: Recentering: $\mathcal{C}_1^{ctr} = T_1(\mathcal{C}_1)$ and $\mathcal{C}_2^{ctr} = T_2(\mathcal{C}_2)$
5: Rescaling: $\mathcal{C}_1^{str} = S_1(\mathcal{C}_1^{ctr})$ and $\mathcal{C}_2^{str} = S_2(\mathcal{C}_2^{ctr})$
6: Rotation: $\mathcal{C}_1^{RPA}$ and $\mathcal{C}_2^{RPA}$
7: Translation of $\mathcal{C}_1$ to have the same center of mass as $\mathcal{C}_2$ (separately for original and manipulated patches): $\mathcal{C}_1^{trg}$
8: Initialization of $\alpha_1 \sim \mathcal{U}[0.1, 0.9]$ and $\alpha_2 \sim \mathcal{U}[0.1, 0.9]$
9: Perform a first sub-optimal interpolation between $\mathcal{C}_1^{RPA}$ and $\mathcal{C}_1^{trg}$ with these random values of α_1 and α_2
10: Only keep most confident samples (see main text) and construct two pseudo-labeled sets: almost surely original and almost surely manipulated testing patches
11: Find optimal interpolation coefficients α_1 and α_2 in Equation 4.14 based on the maximization of the sum of log-likelihood on the two sets of pseudo-labeled patches: $\mathcal{C}_{adp}^{ori}$ and $\mathcal{C}_{adp}^{mnp}$
12: Repeat steps 8 to 11 for five times and keep the adapted GMMs with highest sum of log-likelihood.
13: **return**

Log-likelihood $\mathcal{LL}_{adp}^{ori}$ is computed on almost surely original patches of the testing set, with adapted covariances and corresponding original weights. Similarly, $\mathcal{LL}_{adp}^{mnp}$ is computed on manipulated pseudo-labeled patches of the testing set with the adapted GMM covariances. Optimal values for α_1 and α_2 depend on the manipulation and resizing factor, so the full interval $[0, 1]$ is searched. Like for the EM algorithm, the performances depend on the initial random values of α_1 and α_2 for the first sub-optimal interpolation. Therefore, initialization is performed multiple times (five times is experimentally a good trade-off between computation time and performances), and the one with highest log-likelihood on pseudo-labeled test samples is selected as the final adapted GMMs. To summarize, in GRAFT, in order to adapt GMMs to the new target domain, simple operations (translation, scaling, rotation and interpolation) are performed to adjust covariance matrices in an unsupervised manner, and the procedure is driven by GMMs likelihood maximization on pseudo-labeled target samples.

Table 4.9 – Testing accuracy (in %) with adaptation of GMMs (resizing $\times 0.51$). The improved accuracy, compared to the setting without adaptation, is given in parentheses.

	GF	MF	USM	WGN	JPEG	AVG
Resizing $\times 0.51$ (without adaptation)	64	75	73	79	79	74
Resizing $\times 0.51$ (retraining with 10%)	77 (+13)	79 (+4)	82 (+9)	84 (+5)	81 (+2)	81 (+7)
Resizing $\times 0.51$ (unsupervised, GRAFT)	79 (+15)	79 (+4)	88 (+15)	89 (+10)	83 (+4)	84 (+10)

Table 4.10 – Testing accuracy (in %) with adaptation of GMMs (resizing $\times 0.71$), with improved accuracy in parentheses.

	GF	MF	USM	WGN	JPEG	AVG
Resizing $\times 0.71$ (without adaptation)	78	81	81	91	84	83
Resizing $\times 0.71$ (retraining with 10%)	82 (+4)	81 (+0)	93 (+12)	93 (+2)	89 (+5)	88 (+5)
Resizing $\times 0.71$ (unsupervised, GRAFT)	79 (+1)	81 (+0)	91 (+10)	92 (+1)	84 (+0)	85 (+2)

Table 4.11 – Testing accuracy (in %) with adaptation of GMMs (resizing $\times 1.25$), with improved accuracy in parentheses.

	GF	MF	USM	WGN	JPEG	AVG
Resizing $\times 1.25$ (without adaptation)	51	75	74	81	67	70
Resizing $\times 1.25$ (retraining with 10%)	70 (+19)	80 (+5)	95 (+21)	89 (+8)	80 (+13)	83 (+13)
Resizing $\times 1.25$ (unsupervised, GRAFT)	83 (+32)	84 (+9)	83 (+9)	96 (+15)	88 (+21)	87 (+17)

4.4.4.2 Results

Table 4.3 presents the testing accuracy, without any adaptation, under different resizing factors for the GMM-based method (the same results as those shown in Fig. 4.1 in Section 4.1), where $\times 1$ stands for the case of original-sized testing images without any resizing pre-processing. We can see that there are obvious accuracy drops for both downscaling and upscaling. Our objective is to improve the accuracy on pre-resized testing data, in an unsupervised manner, by using the proposed GRAFT algorithm.

1) With fixed pre-resizing factor

The results of our approach GRAFT are presented in the last row of Table 4.9 (pre-resizing factor of 0.51), Table 4.10 (pre-resizing factor of 0.71) and Table 4.11 (pre-resizing factor of 1.25, upscaling). We also show results of another method: retraining new GMMs from scratch by using 10% testing samples with ground-truth labels. It corresponds to a scenario where few target pre-resized data are available with labels. This scenario is also a good indicator of the difficulty of the task. Indeed, if a detector trained with few labeled target data performs bad on target domain, the problem should be difficult. From the tables, we can see that GRAFT method gives performance improvement in all cases, except for two situations, *i.e.* MF and JPEG under resizing of 0.71. However, in the first case of MF, even the retraining method gives no improvement, which implies that the forensics problem becomes more difficult for the GMM-based method. In addition, under many testing scenarios there is no big performance gap between unsupervised GRAFT and the retraining method which does use ground-truth labels from the target domain. In certain cases, retraining GMMs can lead to big accuracy increase when compared with GRAFT (*e.g.*, JPEG with resizing of 0.71), though at the expense of higher computational cost and with the assumption of gaining access to true labels.

The improved average accuracy of GRAFT (the last figure at the bottom right in Tables 4.9, 4.10 and 4.11) is quite satisfying, around 85%. For resizing factor of 0.51 (Table 4.9), we are able to gain in average $+10\%$ of testing accuracy with the GRAFT method. The performance improvement of GRAFT depends on the manipulation operation, ranging from the smallest increase of $+4\%$ for MF and JPEG to the biggest increase of $+15\%$ for GF and USM. The improvement is to some extent correlated with the drop in performances induced by pre-resizing. For the resizing factor of 0.71 (Table 4.10), we can see that even with the retraining method only limited improvements can be achieved. This case is more difficult than resizing $\times 0.51$ or $\times 1.25$, which explains the moderate gains in performance with our GRAFT method, though the average accuracy after adaptation remains satisfying. Our method is not limited to downscaling and also provides good results with upscaling, as shown in Table 4.11 with the case of the resizing factor of 1.25. A considerable average accuracy improvement of $+17\%$ is achieved by our GRAFT method. At last, it can be observed from Table 4.9 and Table 4.11 that sometimes GRAFT can achieve higher accuracy than the retraining method. One possible explanation is that the knowledge gained from original-sized source domain is beneficial to improve the performance of the same task in the new pre-resized domain.

It is also interesting to see some cases where GRAFT obtains even slightly higher accuracy than the results given in Table 4.5 of Section 4.3.2 where GMMs are retrained on a large number of 800000 pre-resized patches. More precisely, for detecting GF and JPEG under resizing

Table 4.12 – Testing accuracy (in %) with adaptation of the GMM-based method by GRAFT for the case of mixed pre-resizing factors. The pre-resizing factors are drawn following the uniform law within the specified interval.

	GF	MF	USM	WGN	JPEG	AVG
Resizing $\times[0.48, 0.72]$ (without adaptation)	71	81	76	72	87	77
Resizing $\times[0.48, 0.72]$ (unsupervised, GRAFT)	80 (+9)	82 (+1)	89 (+13)	77 (+5)	89 (+2)	83 (+6)
Resizing $\times[1.12, 1.27]$ (without adaptation)	53	78	81	83	74	74
Resizing $\times[1.12, 1.27]$ (unsupervised, GRAFT)	63 (+10)	79 (+1)	89 (+8)	95 (+12)	76 (+2)	80 (+6)

of $\times 1.25$, GRAFT achieves 83% and 88% while training GMMs with 800000 patches of pre-resized images gives 75% and 85%, respectively. One possible explanation is that the knowledge gained on original-sized images is helpful to boost the performance of the same task in another domain of pre-resized images. We can also notice that the performance in Table 4.5 is slightly different from the result of original-sized images in the row of $\times 1$ in Table 4.3. This means that the difficulty of detecting the considered manipulations slightly changes when we change the statistics of images that are considered as original.

2) With a mix of factors

In Table 4.12, the target data are not pre-resized with a specific factor but a random factor is drawn in some interval. As depicted in the table, the GRAFT approach is able to cope with a testing set composed of a mix of pre-resizing factors. It is a more difficult problem as the target domain is more diverse. Hence, performance gain in average is $+6\%$. An explanation is that our method relies on likelihood maximization and with a mix of factors, likelihood is optimized on average. Therefore when testing with a patch at a specific factor it is not as optimal as previously. Yet our method remains useful in particular for unsharp masking (USM) detection, and the average accuracy after the improvement of GRAFT is higher than or equal to 80% for both cases.

3) Illustration of the effectiveness of interpolation

In order to show the effectiveness of interpolation of covariance matrices in GRAFT, Figure 4.6 illustrates the evolution of the testing accuracy as a function of the interpolation coefficients α_1 and α_2. For this case of adapting a detector of GF to a resizing of $\times 0.51$ as pre-processing, the derived optimal values are around 0.2 for both α_1 and α_2. The upper right corner $(1, 1)$ corresponds to C^{RPA} and the lower left corner $(0, 0)$ to C^{trg}. None of these two points are

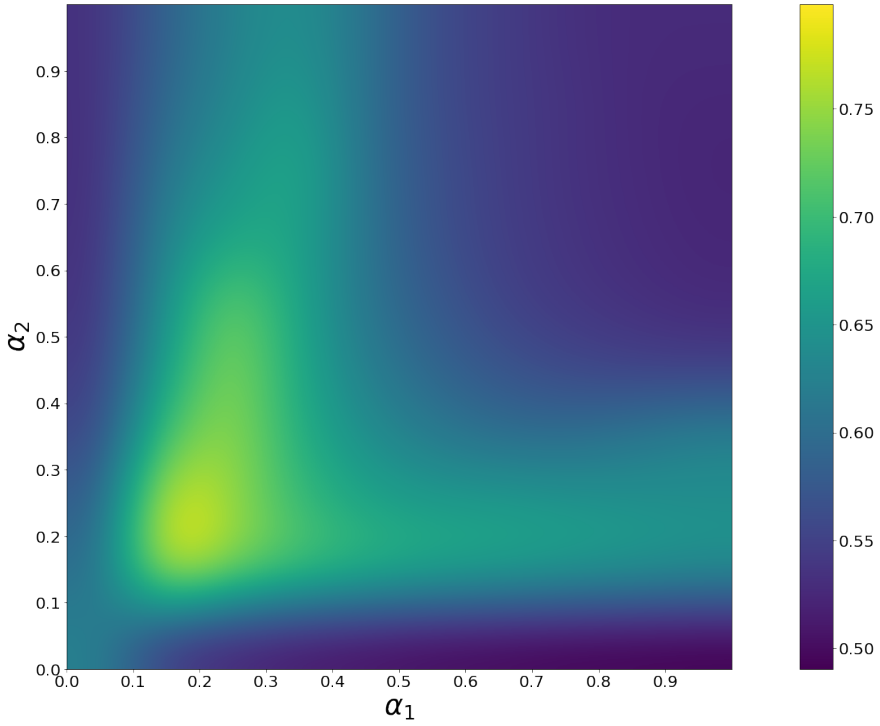


Figure 4.6 – Testing accuracy as a function of the interpolation coefficients α_1 and α_2. The manipulation is Gaussian blurring and the pre-resizing factor is 0.51.

optimal in terms of accuracy or likelihood, and the interpolation between the two can improve the detection performance.

4) Computation power

At last, we would like to mention that our method is much less time consuming than the 150 iterations of the EM algorithm for the GMM training. We were actually looking for shortcuts to perform lightweight and flexible adaptation as in the proposed GRAFT algorithm. Regarding computation time, on our machine with Intel Xeon E5-2630 CPU, it takes approximately 15min to compute the GRAFT procedure, while it needs about 5h to train one GMM with 400000 samples. The weakly-supervised GMM adaptation (Section 4.4.3) takes about 3min (1min for the optimization problem and 2min for DNN fine-tuning). Training of the CNN-based method of Bayar and Stamm [BS16] (Section 4.3.1) takes around 2h for each binary problem on an Nvidia 1080 Ti GPU (it would be much longer on a CPU). Finally, the extraction of SPAM features [PBF10] (Section 4.4.1) is quite fast (about 1h) as it can be done in parallel on the 40 cores of our CPU. However, classifier training is about 10h as the feature dimension is very large.

Table 4.13 – Testing accuracy (in %) with adaptation of GMMs in the case of nearest-neighbor interpolation used for carrying out resizing pre-processing. The improved accuracy, compared to the setting without adaptation, is given in parentheses.

	GF	MF	USM	WGN	JPEG	AVG
Without resizing	91	86	97	98	89	92
Resizing $\times 0.51$ (without adaptation)	66	79	82	80	79	77
Resizing $\times 0.51$ (unsupervised, GRAFT)	78 (+12)	84 (+5)	90 (+8)	84 (+4)	79 (+0)	83 (+6)
Resizing $\times 0.71$ (without adaptation)	76	79	82	80	80	79
Resizing $\times 0.71$ (unsupervised, GRAFT)	82 (+6)	82 (+3)	93 (+11)	84 (+4)	83 (+3)	85 (+6)
Resizing $\times 1.25$ (without adaptation)	80	80	90	91	79	84
Resizing $\times 1.25$ (unsupervised, GRAFT)	79 (-1)	82 (+2)	95 (+5)	93 (+2)	80 (+1)	86 (+2)

4.4.4.3 Additional results

We provide here some more results for the GRAFT procedure on two different pre-processing and development pipelines: pre-resizing with nearest-neighbor interpolation and JPEG compression. The study is not as exhaustive as the study on pre-resizing with bi-cubic interpolation. The objective is to illustrate other experimental scenarios.

1) Nearest-neighbor interpolation

The results reported in Table 4.13 are somewhat similar to those obtained with bi-cubic interpolation shown previously in the manuscript. The same trends are observed. It can be seen that in general nearest-neighbor interpolation induces less performance decrease than the more challenging bi-cubic interpolation, which is understandable. In particular, under upscaling the performance drop induced by nearest-neighbor interpolation is noticeably smaller, and in accordance the improvements of GRAFT are smaller too. The final average improved accuracy after adaptation for different factors in Table 4.13 is very similar to the value obtained with bi-cubic interpolation (see Table 4.3), both are around 85%.

2) Pre-processing with JPEG

In this set of experiments, we consider another image development pipeline. JPEG images are now considered as original images and we try to detect various manipulations applied on

Table 4.14 – Testing accuracy (in %) with adaptation of GMMs to pre-resizing of $\times 0.51$ and $\times 1.25$. Here we consider JPEG images as original ones and try to detect manipulations applied on them. The improved accuracy, compared to the setting without adaptation, is given in parentheses.

	GF	MF	USM	WGN	JPEG	AVG
Without resizing	72	84	92	90	79	83
Resizing $\times 0.51$ (without adaptation)	71	80	81	70	71	75
Resizing $\times 0.51$ (unsupervised, GRAFT)	74 (+3)	80 (+0)	90 (+9)	87 (+17)	67 (-4)	80 (+5)
Resizing $\times 1.25$ (without adaptation)	58	80	81	90	62	74
Resizing $\times 1.25$ (unsupervised, GRAFT)	65 (+7)	80 (+0)	95 (+14)	96 (+6)	63 (+1)	80 (+6)

Table 4.15 – Testing accuracy (in %) of CNN-based method with and without pre-resizing. We consider JPEG images as original ones and try to detect manipulations applied on them.

	GF	MF	USM	WGN	JPEG	AVG
Without resizing	75	87	90	86	65	81
Resizing $\times 0.51$	50	53	52	82	60	59
Resizing $\times 0.71$	51	64	60	84	61	64
Resizing $\times 0.91$	55	75	75	86	65	71
Resizing $\times 1.25$	67	82	88	80	66	77

them. Images of the Dresden database are JPEG-compressed with a quality factor randomly selected in $[91, 92, 93, 94, 95, 96, 97, 98, 99]$. JPEG images are then pre-resized (or not) and finally manipulated (or not) to generate different training and testing datasets. This setting has been tested for GMM and CNN. As can be observed in Table 4.14 and Table 4.15, base scores without pre-resizing are lower, which means that the forensics problem of detecting manipulations becomes more difficult on JPEG images considered as original ones. Of course, JPEG $Q = 90$ detection is way more difficult as some images are labeled as original and compressed with $Q = 91$. The performance drops under pre-resizing can be observed for both GMM- and CNN-based methods and adaptation seems more difficult in this case. Nevertheless, GRAFT is still able to restore some performances, and the improved average accuracy (80% for both scaling factors in Table 4.14) is rather satisfactory and is not far away from the average base score on patches of original-sized images (83% as presented in Table 4.14).

Table 4.16 – Testing accuracy (in %) of the weakly-supervised method from [DWC19b] (presented in Section 4.4.3 of this manuscript) and its unsupervised variant, with comparisons with GRAFT. "sup." and "unsup." stand respectively for "supervised" and "unsupervised".

	GF	MF	USM	WGN	JPEG	AVG
Resizing $\times 0.51$ (weakly sup. of [DWC19b])	78	76	92	70	86	80
Resizing $\times 0.51$ (unsup. version of [DWC19b])	71	75	82	65	84	75
Resizing $\times 0.51$ (unsupervised, GRAFT)	79	79	88	89	83	84
Resizing $\times 1.25$ (weakly sup. of [DWC19b])	66	80	95	95	78	83
Resizing $\times 1.25$ (unsup. version of [DWC19b])	64	75	91	92	68	78
Resizing $\times 1.25$ (unsupervised, GRAFT)	83	84	83	96	88	87

4.4.5 Comparisons of different approaches

We compare in this section the fine-tuning of Bayar and Stamm's CNN [BS16], the weakly-supervised adaptation of GMMs (Section 4.4.3) and the fully-unsupervised adaptation of GMMs named GRAFT (Section 4.4.4).

4.4.5.1 Comparisons of likelihood-based approaches

The method described in Section 4.4.3 requires some labels on the target domain, while the GRAFT method (Section 4.4.4) performs unsupervised adaptation. The results of weakly-supervised method and unsupervised methods are presented side-by-side in Table 4.16. The row of "weakly sup. of [DWC19b]" corresponds to the original method described in Section 4.4.3. It can be observed that the GRAFT method achieves comparable or even slightly better performances though considering a more challenging setting of unsupervised adaptation. For instance, the average accuracy after applying weakly-supervised method with a pre-resizing factor of $\times 0.51$ is 80% while for the GRAFT method it is 84%. The average accuracy for upscaling of $\times 1.25$ is also in favor of GRAFT with 87% compared to 83% for the weakly-supervised adaptation.

For further comparisons, instead of using few labels as in the original weakly-supervised

method for adaptation, we have implemented an unsupervised version that uses pseudo-labels instead. Pseudo-labeled samples are determined similarly as in the GRAFT method (Section 4.4.4, 3) Pseudo-labels). The results of this unsupervised variant are presented in Table 4.16, on the row of "unsup. version of [DWC19b]". The unsupervised version has lower accuracy than the original weakly-supervised version, which is understandable because less information on the target domain is available. The GRAFT method outperforms the unsupervised variant of [DWC19b].

The better performance of GRAFT may be due to the formulation of the adaptation as a maximization of GMM likelihood, instead of only considering the fit of covariance statistics. Both methods use, directly or indirectly (through responsibilities), a comparison of log-likelihood of patches under the two GMMs. Therefore, the GRAFT method which directly maximizes log-likelihood on the target data provides better results than the weakly-supervised method which aims to indirectly increase the log-likelihood on the target data by a better covariance fitting.

In Figure 4.7 we present a toy example of an image falsification in which a manipulation operation is involved. We have spliced a JPEG-compressed ($Q = 90$) part into an uncompressed image. The JPEG compression is detected by the conventional GMM detector in Figure 4.8 which allows us to localize the falsification. Figure 4.9 is the output of the same GMM detector when the source image and spliced part have been resized before manipulation and splicing. The falsification is not accurately localized anymore because of the performance drop for manipulation detection. Finally in Figure 4.10, results after the weakly-supervised and the GRAFT adaptation procedures are depicted. Thanks to the enhancement of performance for manipulation detection provided by these two procedures, the spliced part is again accurately localized. Adaptation is performed specifically for this single image, only with patches from this image. As the the weakly-supervised approach uses label information for the very image under test, it achieves a better localization compared to the GRAFT procedure, in contrast to the results in Table 4.16.

4.4.5.2 Comparisons with CNN-based method

As described in Section 4.4.2, we have modified Bayar and Stamm's CNN [BS16] so that the network is now able to detect manipulations on 8×8 patches. We would like to emphasize here that the manipulations considered here are more challenging than in their paper [BS16] (smaller magnitude, smaller kernel for median filtering, *etc.*) and that patches are much smaller. Therefore performances of the network are reduced compared to the performances reported in Bayar and Stamm's paper [BS16].

Figure 4.7 – Splicing of a JPEG $Q = 90$ part (within the red circle) into an uncompressed host image.

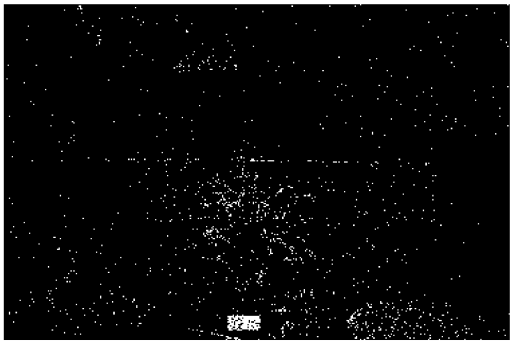


Figure 4.8 – Result on original-sized image.

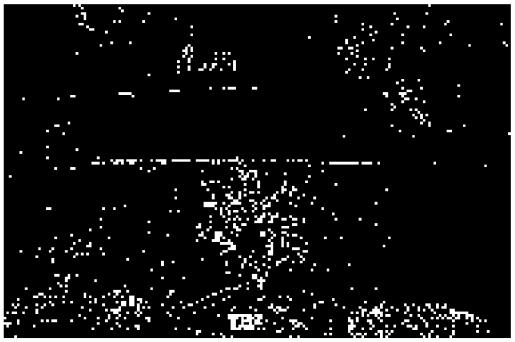
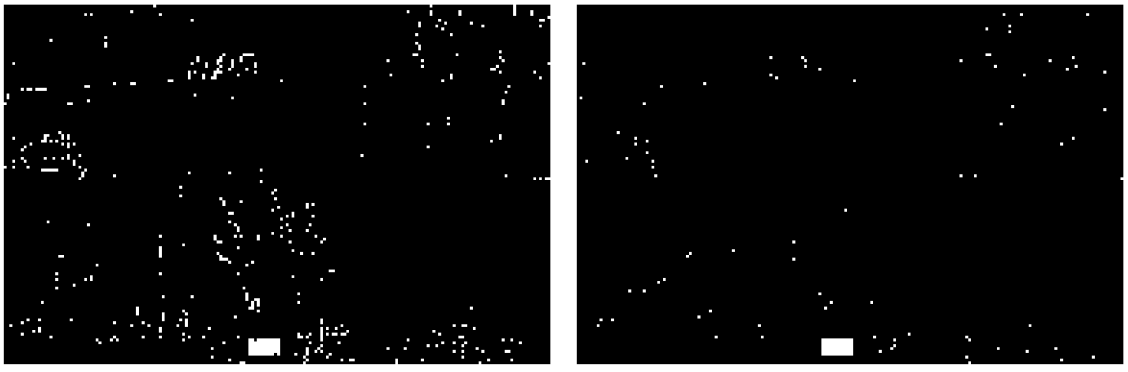


Figure 4.9 – Result without adaptation on prerezised image with scaling factor $\times 0.51$.



(a) (b)

Figure 4.10 – Results obtained after applying our adaptation algorithms. (a) After GRAFT adaptation; (b) After weakly-supervised adaptation.

The accuracy of the CNN-based method is presented in Table 4.6. The CNN's base performance without resizing is lower than that of the original GMM [FWC15], -8% in average (84% for CNN *vs.* 92% for GMM). The CNN also suffers from an accuracy drop under pre-resizing. In general, although CNN has a smaller accuracy drop (this point deserves further studies), but the final decreased accuracy without adaptation is comparable for CNN-based and GMM-based methods (*cf.* the corresponding rows in Tables 4.6 and 4.7), with a same trend of more accuracy drop under upsampling for both methods. In addition, GMM adapted with GRAFT procedure (the best of the two likelihood procedures, see Section 4.4.5.1) achieves after adaptation to pre-resizing average accuracy values (around 85%) comparable to the base average score of the CNN when there is no pre-resizing. CNN-based methods have deeply transformed the image forensics field (and computer vision field in general) and provide usually significantly better results on classical benchmarks. However, it seems that in this particular framework that deals with small patches, the classical approach remains competitive and offers even better performances. This result reminds us that even though the CNN-based methods are very promising and therefore should be investigated first and foremost, it is still valuable to explore classical methods for scenarios where CNN-based methods are limited.

We have made efforts to carry out unsupervised adaptation to pre-resizing for CNN-based detectors (to our knowledge no such methods exist in the image forensics literature). We tried to use pseudo-labels but without success. By design, CNN-based classifiers are learned through *back-propagation*. It computes the error between the true and the predicted labels. With pseudo-labeled samples, as a CNN is deterministic, there are no differences between the labels and the prediction; therefore, there is nothing to learn for the CNN in this case. Some methods make use of pseudo-labeled samples [Lee13]; [Zha+18] but always combined with true labels. By contrary, the GMM's descriptive capability for pseudo-labeled samples can be effectively improved, which leads to actual performance improvement.

Beside that, we have fine-tuned the CNN-based method of [BS16] with a limited number of samples (2000) as described in Section 4.4.2, which corresponds to the weakly-supervised scenario of Section 4.4.3. Therefore, the results of this fine-tuning are compared with results of the weakly-supervised adaptation of GMMs, in Table 4.17. Our method provides better results on average than fine-tuning Bayar and Stamm's CNN with a limited number of samples. It reflects here that CNNs need more sample for training and fine-tuning than our method. It was our aim to be able to provide such additional flexibility. Only for median filtering, CNN results exceed results of our method. It seems quite logical, as the median filtering is the weakness of our method. In fact and as depicted in Table 4.7, our method is not able to recover more than 5 percent of performance for median filtering, while for other manipulations

Table 4.17 – Testing accuracy (in %) for our adapted GMM-based method (weakly-supervised scenario) and fine-tuned CNN-based method. The training set contains original-sized images while the testing set contains pre-resized images of a particular resizing factor. The improved accuracy of the best method compared to the other one is given in parentheses.

	GF	MF	USM	WGN	JPEG	AVG
Resizing $\times 0.51$ (our method)	**78 (+5)**	76	**92 (+7)**	**70 (+1)**	**86 (+9)**	**80 (+3)**
Resizing $\times 0.51$ (CNN fine-tuning)	73	**82 (+6)**	85	69	77	77
Resizing $\times 0.76$ (our method)	**83 (+9)**	82	**94 (+5)**	**85 (+5)**	**84 (+5)**	**86 (+5)**
Resizing $\times 0.76$ (CNN fine-tuning)	74	**83 (+1)**	89	80	79	81
Resizing $\times 1.15$ (our method)	**70 (+3)**	**85 (+5)**	**95 (+4)**	**96 (+10)**	**82 (+16)**	**86 (+8)**
Resizing $\times 1.15$ (CNN fine-tuning)	67	80	90	86	66	78
Resizing $\times 1.25$ (our method)	**66 (+1)**	**80 (+3)**	**95 (+4)**	**95 (+9)**	**78 (+18)**	**83 (+7)**
Resizing $\times 1.25$ (CNN fine-tuning)	65	77	91	86	60	76

the improvement range is up to more than 10 percent.

4.5 Summary

In this chapter, we have tackled the problem of image manipulation detection. The performances of several state-of-the-art methods have been studied when the testing set is exposed to resizing as pre-processing, before the addition of a manipulation. The aim was to investigate the generalization power of existing detectors in a realistic scenario. In this scenario, images would have been pre-processed, resized, because of storage limitations or for layout purposes before being retrieved to be tested. Detectors would have been trained on a regular database of full-sized images. We show that in such a scenario, drops of performances are observed for detectors of three different types: a CNN-based [BS16], another one based on SPAM-features [PBF10] and a last one based on statistical modeling of patches with Gaussian Mixture Model (GMM) [FWC15]. We have formally studied the statistical differences of the pre-resized and full-sized manipulated data with a statistical test based on the MMD distance.

We have then introduced three methods to adapt and recover some performance: (i) the classical neural network fine-tuning, (ii) a weakly supervised approach for the GMM-based pipeline and (iii) an unsupervised adaptation approach named GRAFT still for the GMM-based pipeline. For (ii) and (iii), flexibility is a particular focus. We then present experimental results on a public database (Dresden with automatic manipulations) for all the three methods and study the re-gain of performances after adaoptation. Methods (ii) and (iii) provide better results than (i). We explain it by the additional agility.

We also have briefly introduced methods for feature adaptation. They seem very promising for future work as they are not specific to a model as in our methods. Beside that, neural networks and especially CNNs have shown really powerful results for most of the image forensics problem. But most existing methods seem to lack the flexibility and agility to adapt to new datasets. It seems to us that a truly promising line of work is to develop new neural network methods that are versatile by design. In classical computer vision, this line of work fall mostly under the categories of *few-shots* and *zero-shot* learning.

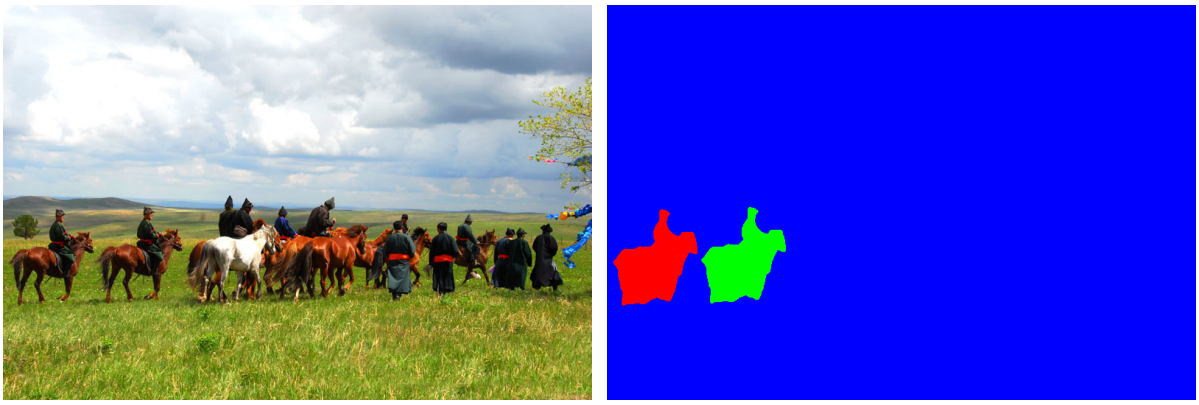
5 | Disentangling Source and Target Areas in Copy-Move

Contents

Influenced by our experience in the DEFALS challenge, we intend to continue our work on falsification detection after the studies on manipulation detection. Our aim is to propose an approach that favours flexibility and adaptability to individual images. The idea is also to take advantage of our experience and knowledge on explicit modeling with statistical models, and especially GMM. Therefore, we choose to focus on copy-move, a popular image falsification where a part of the image, the *source* area, is copied on the *target* area, in the same image.

Most existing copy-move detectors search for matching areas. Thus, they identify the two zones indifferently, while only the target really represents a tampered area. We present a method that automatically distinguishes between the two, to locate the source and the target. This source and target identification is helpful to better understand the semantics of the tampered and original images and can facilitate the work of human forensic investigators. For instance in Figure 5.1, one of the rider has been copied probably to make a bigger group. One could be interested to know what the group really looked like and thus identify the source and target areas. This kind of inferred information about the real position of a person in the original image may be very important, especially for images used in court. For the image shown in Figure 5.1, it is difficult to visually identify the original rider and the tampered one. In addition, the task of visual inspection will be very tedious and time-consuming when there are a lot of suspicious images to be examined. Therefore, it is interesting and necessary to design automatic forensic methods to disentangle the two kinds of areas. To the best of our knowledge, there is only one published method called BusterNet [WAAN18] that is capable of such source and target localization, along with another from a very recent preprint [BPT19], while there are plenty of published and very efficient methods for copy-move detection in which source and target are identified indifferently. Thus, we choose to focus on the disentangling of copy-moved source and target area, rather than copy-move detection. Existing copy-move detectors would be considered as available black boxes in a first stage. We have developed a method that acts as a second-stage detector to identify only the tampered areas in copy-move.

In this Chapter, we first briefly present existing methods for copy-move detection, classifying them as classical feature-based methods and deep-learning-based methods. We then present in detail our proposed method with Gaussian Mixture Models (GMMs). We show some results and comparisons with BusterNet [WAAN18] on copy-move images from CASIA2 [DWT13] and CoMoFoD [Tra+13] datasets. Then we introduce succinctly a variation of our method with another deep-learning-based modeling tool called PixelCNN++. Some experimental results on the same datasets are also provided. Finally comparisons of the existing methods and our two variations are presented.



(a) (b)

Figure 5.1 – Example of copy-move image from CASIA2 [DWT13] database. (a) The rider on the left has been copied. (b) Mask of the tampering, where green indicates the source area and red the target area of the copy-move.

5.1 Copy-move detectors

Many approaches have been developed for copy-move detection. The most recent ones are naturally based on *deep-learning*. As we will show in this chapter, classical *feature-based* methods are still competitive. As introduced earlier, such methods can be used as detectors of the copy-moved zones to then be disentangled with our proposed method. More than two zones can be detected by these detectors, *e.g.,* in the case of multiple copy-moves within the image, false alarms of the detectors, or detection of fragmented copy-moved areas.

5.1.1 Feature-based approaches

There are in general two types of feature-based approaches, extracting features respectively from image blocks and key-points [Chr+12]. For the block-based methods, images are divided into blocks, *i.e.,* overlapping patches. Features are then extracted from these patches and matched across the image. Finally, a step of post-processing is often performed to reduce false alarms. Early features used for copy-move detection were based on Discrete Cosine Transform (DCT) on image patches [FSL03]. Other features were then considered such as Zernike Moments [RLL10]; [CPV15a] and the Tetrolet transform [MT20a]. They are to some extend invariant to scaling and rotation of the copied area, and thus they can provide more robustness to affine transformations than features based on the DCT. Following this feature extraction stage, a matching of blocks is performed. With a naive, exhaustive search, the cost of matching would be cubic in the number of patches. It would quickly become

unpractical with regular sized images or with too much overlap of the patches. Therefore, strategies of approximate nearest-neighbor search have been used in copy-move detectors, such as the PatchMatch [Bar+09] algorithm used by the method of [CPV15a] or the kd-tree used by [LG06]. The last stage of post-processing usually consists of filtering small areas and areas that are too close to each other. Morphological operations may also be performed.

To reduce the cost of nearest-neighbor searching in block-based methods, it is possible to consider only key-points and not the full image. Features are extracted but only on key-points. The following steps of matching and post-processing are then quite similar but with less samples. Typical key-points used for copy-move detection are based on *Scale-Invariant Feature Transform* (SIFT) [PL11]; [MT20b] or *Speeded Up Robust Features* (SURF) [SB11]. They are, by construction, scale-invariant and therefore provide some robustness against resizing. The main drawback of these methods is that most key-points are usually to be found around high entropy region of image. Thus, a copy-move performed in a uniform area, *i.e.,* an area without key-points, would be hardly detected.

5.1.2 Deep-learning-based approaches

Beside approaches based on matching features extracted from either blocks or key-points, a method named BusterNet [WAAN18] has been proposed which employs a *Convolutional Neural Network* (CNN) for copy-move detection. The network is composed of two branches: *Mani-Det* and *Simi-Det*. The former is trained to detect manipulations on the target area, while the latter aims to detect similar copy-moved areas in an image by using a well-designed self-correlation module. Both branches output a heatmap of the same size as the input image. A fusion of the two branches is finally performed. The source and target discrimination in BusterNet is based on the assumption that the target area has been manipulated, *e.g.,* scaled or rotated. Regarding the implementation, branches are first trained separately. This separated training is performed on a synthetic dataset of manipulated copy-move images composed of 100000 samples. The network is quite large so there is no existing realistic dataset that is big enough to train on. In addition, the training requires the use of ground-truth information of source and target in the copy-move which is not always available. The *Mani-Det* branch is trained also with realistic image falsification datasets which are of relatively small scale and typically include a few hundreds or thousands tampered images (IEEE IFS-TC dataset and Wild Web dataset [ZPK15]). Then the fusion module is trained alone, with both branches frozen, and finally the full network is fine-tuned in an end-to-end manner.

The authors of [ZP19] have proposed to use DenseNet instead of a CNN to detect copy-move tampering. Their method also relies on a module of self-correlation computation. How-

ever, unlike BusterNet, this network is not able to identify the source and target areas. Alternatively, the authors of BusterNet proposed to use a CNN initially designed for splicing detection and localization [WAAN17] to expose copy-move forgeries. With some adaptations of the network for training and testing, the adapted CNN can detect well duplicated areas but is not able to distinguish between the source and the target. Recently, the authors of an arXiv pre-print [BPT19] proposed a multi-branch CNN for source and target disambiguation. It is composed of a first branch that compares inside areas of two duplicated zones and a second one comparing boundaries of the two zones.

5.2 Disentangling copy-moved source and target areas with GMM

As explained earlier, most existing methods are powerful in identifying the duplicated areas but they are not able to perform source/target discrimination. Therefore, we intend to develop a new detector which first makes use of such an efficient method to produce a mask of duplicated areas, and then we disentangles the source and the target. In this section, we present our GMM-based method. In the next section, we introduce a variant of our method which makes use of a deep-learning architecture called PixelCNN++.

5.2.1 Overview of the method

A graphical overview of the main steps of our method is shown in Figure 5.2. The inputs of our method are the image to be analyzed and the binary mask produced by a first-stage detector in which duplicated areas are identified indifferently. To distinguish between source and target areas, we first extract patches from the identified pristine region (the black region in the binary mask) and train a GMM to represent the statistics of pristine patches. The next step is to construct the empirical histogram of the patches log-likelihood of all the connected components (CCs) in the binary mask, including the pristine region as well as the candidate source/target areas (the white CCs in the binary mask). We then compute the *intersection* between the histogram of pristine region and that of each candidate source/target area. At last, we consider the candidate area with the largest intersection as the source area and the other(s) as the target area(s).

One important intuition motivating the design of our method is that the boundaries of the target area would expose locally some slight statistical deviations from the rest of image. The authors of BusterNet [WAAN18] make the assumption that the target area has been manipulated and they rely on manipulation detection to discriminate the two zones. We make a

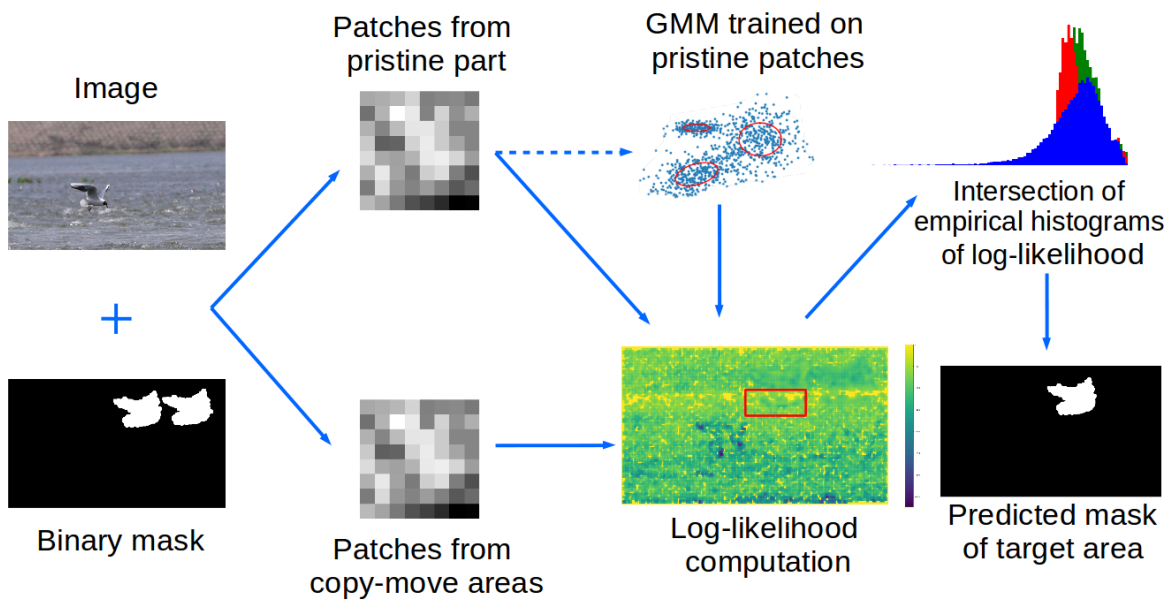


Figure 5.2 – Graphical summary of the main steps of our method to disentangle copy-moved source and target areas. The red histogram of the target area, which is correctly identified in the final mask, has less intersection with the blue histogram of the pristine part than with the source area (green histogram). This is also somehow partially reflected by the small blue band close to the boundary of target area in the log-likelihood map, as highlighted by the red rectangle (zoom-in of the digital version if this manuscript is recommended).

weaker and more general assumption which measures the statistical deviation of local image patches. We believe that by comparing the likelihoods of duplicated areas, we are able to distinguish the source and target areas. First, assuming no manipulation on the target area, the log-likelihood values for interior patches of copy-moved areas would intersect largely with the values of the pristine region, whereas log-likelihood values on boundaries for the target area would deviate. Further, when the target area is manipulated, the statistical deviation would be even bigger for patches from the whole target area, including both interiors and boundaries. So, the area with the log-likelihood values closest to the pristine area is considered as the original (source) one and the one that deviates most would be the target.

Our approach relies on a Gaussian Mixture Model because it is a very good candidate for modelling natural image statistics [ZW11]. It is inspired by the one used for image manipulation detection in our previous work [FWC15]. In this chapter, a single GMM, instead of two GMMs in [FWC15], is trained on 8×8 centered patches (the DC component is removed, resulting in floating-point numbers) coming from a specific image. The patches are extracted from the identified pristine region outside of the duplicated areas as predicated by

the first-stage copy-move detector. The GMM is trained by using the EM algorithm which maximizes the model's log-likelihood of Equation 3.4 on the pristine patches. Therefore, this GMM should capture the regular local statistics of the image. In our work, the GMM is simpler and has less parameters (25 components) compared to the 200 components used for manipulation detection in [FWC15]. Experimentally, it is not necessary to have a very high capacity of description because now a single image is considered and not a big image database as in [FWC15]; [ZW11]. Each GMM component has a full covariance matrix as we intend to capture the subtle dependence in patches, but not only a simplified dependence as diagonal covariance matrices would do.

The idea is then to measure the statistical deviation of each duplicated area from the regular statistics of the image as described by the GMM trained on pristine patches. In practice, we compute and compare the distance between empirical distribution of the log-likelihood on patches in each candidate source/target area and the distribution in the identified pristine part. We propose to use a simple but effective measure of distance based on the number of elements in the intersection of two normalized histograms. To the best of our knowledge, this intersection-based metric was first proposed in [IB89] with the name of *overlapping coefficient* because it attempts to measure the overlapping between two distributions. The metric is later used in various applications, for instance the well-known color-based image indexing method of [SB91]. Concretely, this histogram intersection metric is defined as:

$$intersect(h^c, h^p) = \sum_{i=1}^{n} \min(h_i^c, h_i^p), \tag{5.1}$$

where h^c is the normalized histogram of patch log-likelihood for one candidate source/target area, h^p is the normalized histogram for the pristine part, and n is the number of bins. As histograms are normalized, it is not necessary to (re)normalize this intersection score which is naturally between 0 and 1. The area with the largest intersection with the predicted pristine part is considered as the source and the other one(s) as emanating from the target area(s). The width of the histogram bins is set automatically to have 75 bins within the range of log-likelihood values of patches of the whole image, from both pristine and copy-moved areas. The number of 75 bins has been set empirically, based on the observation of histograms like in the last row of Figure 5.3. The method is not very sensitive to this number, a larger number of bins does not significantly improve the results, while using a rough histogram with less than 50 bins starts decreasing the performance. It is worth mentioning that there are a lot of metrics for measuring the distance between histograms, or between distributions in a large sense: Bhattacharyya distance, Kolmogorov-Smirnov statistic, Kullback-Leibler divergence, Hellinger distance, and Chi-Squared distance, just to name a few. The intersection-based distance of Equation 5.1, which is used in our method, is probably one of the simplest metrics.

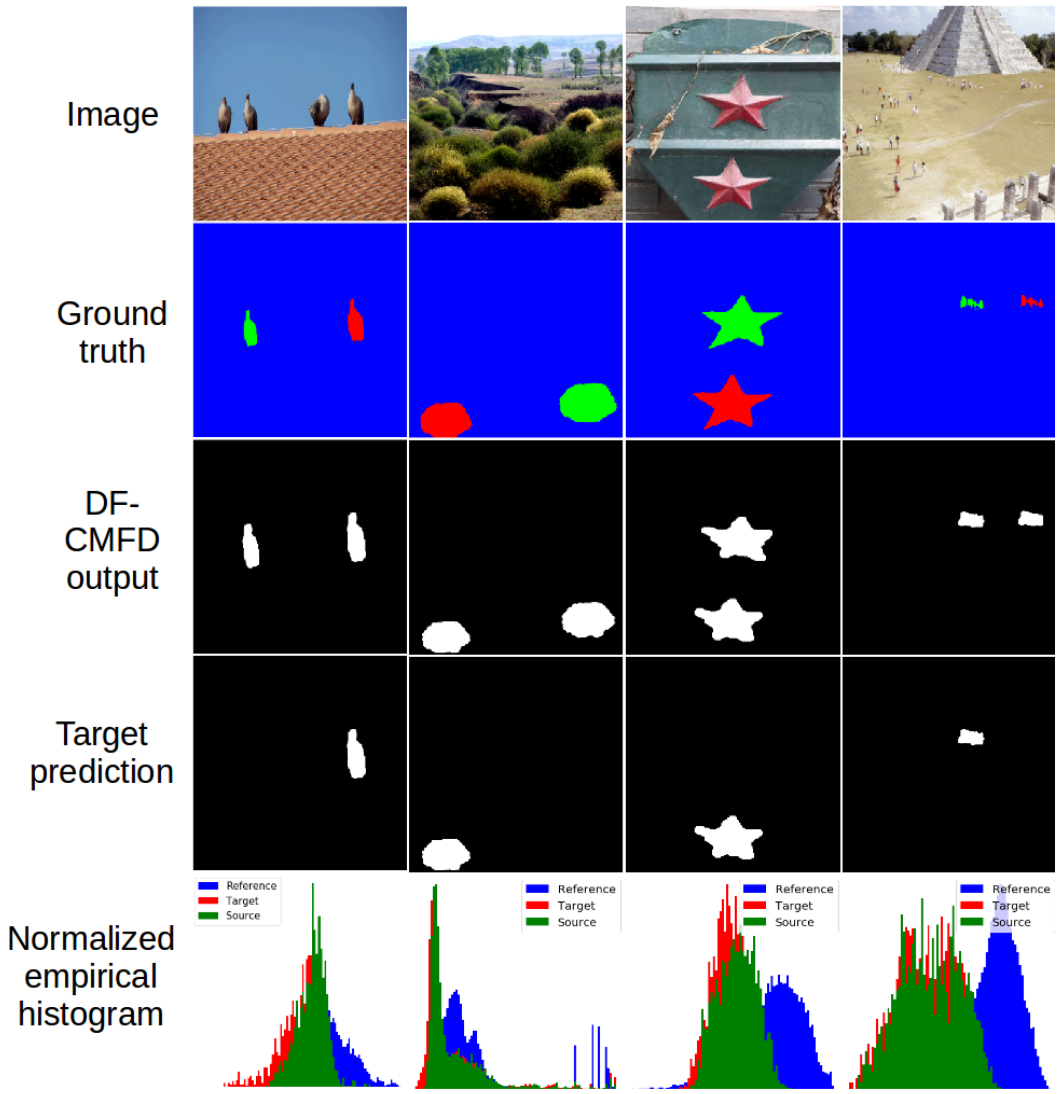


Figure 5.3 – Example results of our method. The red color in ground-truth masks indicates the target area, the green color indicates the source area, and the blue color indicates the pristine part. The example in the last column is a miss-classification.

We have explored empirically two other measures, namely *(i)* Kolmogorov-Smirnov statistic and *(ii)* Chi-Squared distance. They do not provide clearly better empirical results but are more complex and more costly to compute. We have also tried to compare empirical normalized histograms through comparisons of straightforward statistics such as the mean or the median. However, these statistics are in some cases not strong enough to capture the differences. Therefore, this metric of histogram intersection appears to be a good technical choice in terms of both simplicity and effectiveness.

Some examples of histograms and output masks of target area on the CASIA2 dataset can

be found in Figure 5.3, with the dense-field copy-move forgery detector (DF-CMFD) [CPV15a] as the first-stage detector. In the last column of Figure 5.3, histograms for the two areas are very close which led to an error of classification. When the target area has been manipulated, for example in the third column a small deformation has been applied according to the specification of the CASIA2 dataset, the difference in histogram intersections can be more visible.

5.2.2 Experiments

5.2.2.1 Metrics and datasets

We designate as "Discernible" all the samples with at least two white CCs in the binary mask as produced by first-stage detector, and these CCs should intersect with the source and target areas in the ground-truth masks. The remaining samples are in the "Indiscernible" subset, as it is not possible for a second-stage detector to disentangle source and target. This subset mainly comprises samples for which the binary mask has no white CC or the CCs do not intersect with the source and target areas in the ground-truth mask. Samples with only one white CC cannot be disentangled neither, so they are also considered as "Indiscernible". To measure the discerning capability of source and target, we compute the overall accuracy and the accuracy on the "Discernible" subset, as the ratio of number of images with correct source/target classification to the number of images in the whole dataset or in the "Discernible" subset.

The authors of BusterNet [WAAN18] consider a subset of images which they call "Opt-Out". It includes samples with final predicted masks in which all duplicated CCs receive a same label (source or target) as given by BusterNet. We consider that labeling all CCs with same label as a miss-classification for discerning source and target. In our setting, such samples are naturally in the "Discernible" subset, and the performance is evaluated on them as well as on other images in the subset. In [WAAN18] the authors also define a "Miss" subset composed of predicted masks that do not intersect with duplicated CCs in the ground-truth mask. This "Miss" subset is included in our "Indiscernible" subset. Finally, in [WAAN18], "Opt-In" is the subset comprising all remaining images except those in the "Miss" and the "Opt-Out" sets. The authors report performance on these "Opt-In" samples. We think that the "Opt-In" samples in [WAAN18] represent a subset where the BusterNet classifier performs the best. Therefore, for a fair comparison, we consider this subset of images as a "*Favorable Subset*". For our method, we compute a ranking to extract images where the classification is the most certain to also obtain a "Favorable Subset". This ranking is based on the descending

order of a score (a kind of Michelson contrast) computed on images with binary masks of the two duplicated zones, as:

$$score = \frac{|intersect(h^{c1}, h^p) - intersect(h^{c2}, h^p)|}{intersect(h^{c1}, h^p) + intersect(h^{c2}, h^p)}, \tag{5.2}$$

where $intersect(.,.)$ is defined in Equation 5.1, and h^{c1}, h^{c2} and h^p are the normalized empirical histogram for respectively the first zone, the second one and the pristine part. A high score would indicate that one duplicated area exposes a big deviation from the pristine background when compared to the other one. It could be that the target area has been manipulated to cover visual clues of copy-move. We set a threshold on the score in order to have the same number as the "Opt-In" images defined by BusterNet, *e.g.,* 190 for the CASIA2 dataset. A ranking is particularly interesting from an operational point of view since it makes possible to determine the most suspicious images among a large dataset. Finally, as expected, images with a high ranking score decorrelate well from the "Indiscernible" images. For instance, on the CASIA2 dataset with the GMM-based method on DF-CMFD masks, in the subset of the 200 images with the highest score, only 5 are in the "Indiscernible" subset. Still from an operational point of view, it could help us to detect miss-detection of the first-stage copy-move detector.

We test our proposed method on copy-move images from the CASIA2 [DWT13] and CoMoFoD [Tra+13] datasets. We use ground-truth masks from the authors of BusterNet [1] for the two datasets. These ground-truth masks provide information on source and target areas of copy-move: the red channel of the masks contains information about the target area, the green channel on the source area and the blue one on the pristine part of images (*cf.,* Figure 5.3).

5.2.2.2 DF-CMFD[CPV15a] as first-stage detector

With the aim to produce a better pipeline for copy-move detection with source and target disentangling, we look for the best possible first-stage copy-move detector available. Scores for some state-of-the-art detectors on the CASIA2 dataset are reported in Table 5.1. The results for [RLL10], [Chr+12] (SIFT-based method) and [WAAN17] are extracted from the BusterNet paper [WAAN18]. The details of the transformation of [WAAN17] from a splicing to a copy-move detector can be found in [WAAN18]. We also report partial results of a recent DenseNet-based method [ZP19] as presented in the method's original paper. We were unable to reproduce the lower scores reported in [WAAN18] for DF-CMFD [CPV15a], but our results (the column of "DF-CMFD – With default HP" in Table 5.1) are coherent with the reported

1. https://github.com/isi-vista/BusterNet

Table 5.1 – Analysis of copy-move detection and localization performance on the CASIA2 dataset (source and target are identified indifferently). "HP" means hyper-parameters. Please refer to [WAAN18] for details of the three evaluation protocols.

	Methods					DF-CMFD [CPV15a]	
	[WAAN18]	[RLL10]	[Chr+12]	[WAAN17]	[ZP19]	**With default HP**	**With tuned HP**
	Image Level Evaluation Protocol						
Precision	78.22%	97.01%	68.49%	66.37%	-	**97.64%**	71.44%
Recall	73.89%	24.47%	67.82%	73.59%	-	56.81%	**88.80%**
F1-score	75.95%	39.08%	68.15%	69.80%	-	71.83%	**79.18%**
	Pixel Level Evaluation Protocol - A						
Precision	77.38%	**94.46%**	64.84%	17.06%	-	86.20%	92.47%
Recall	59.15%	25.05%	0.17%	10.60%	-	64.43%	**65.85%**
F1-score	67.05%	39.59%	0.34%	13.08%	-	71.62%	**76.92%**
	Pixel Level Evaluation Protocol - B						
Precision	55.71%	22.71%	37.09%	23.97%	**70.85%**	48.18%	67.32%
Recall	43.83%	13.36%	0.14%	13.79%	58.85%	49.40%	**73.93%**
F1-score	45.56%	16.40%	0.23%	14.64%	64.29%	47.88%	**67.97%**

performance of DF-CMFD in [ZP19] (Table III, column of "PM"). We have used the original implementation of DF-CMFD by the authors which is available online [2].

To perform source/target discrimination, it is obviously better to have a detector that correctly identifies a large number of copy-move images. This is mainly reflected in Table 5.1 by the F1-scores at "Image Level" and under the "Pixel Level Evaluation Protocol A". Following this criterion, DF-CMFD [CPV15a] appears to be a good candidate for a first-stage detector. However, one drawback is that the default parameters of DF-CMFD are better suited for big images as considered in the original paper [CPV15a]. For instance, the minimum size considered for clones is 1200 pixels. This represents quite a big area for images of sizes from 240×160 to 900×600 pixels of the CASIA2 dataset. The features are extracted in circular regions of diameter 16 pixels, which is also large for small images. Therefore, this setting is unfavorable for our second-stage detector because predicted copy-move images and masks are not very accurate. As reported in Table 5.1, the F1-score for "Pixel Level Evaluation Protocol B" (localization performance on images with copy-move forgery) for the CASIA2 dataset is only of 47.88%. Therefore, to obtain a higher score, it is better to tune the hyper-parameters of DF-CMFD.

We tune empirically the hyper-parameters with regard to the size of images. We reduce the feature size, the minimum size of clones and distance between clones. For instance for CASIA2, we have empirically selected a feature size of 12, a minimum size of 225 pixels and

2. http://www.grip.unina.it/research/83-multimedia_forensics/90-copy-move-forgery.html (last accessed: Sept. 2020).

Table 5.2 – Source/target discernibility performances of our GMM-based method, BusterNet [WAAN18] and multi-branch CNN [BPT19]. "Indisc." means the "Indiscernible" subset, "Disc." the "Discernible" subset, "Corr." are images with correct classification of source and target, and "Fav. Sub." is the "Favorable Subset". Please refer to Section 5.2.2.1 for details on the evaluation metrics used in the table.

		Number of images					Accuracy		
	Dataset	Total	Indisc.	Disc.	Corr.	Corr. in Fav. Sub.	Overall	Disc.	Fav. Sub.
Our method with tuned DF-CMFD	CASIA2	1313	489	824	549	149	41.81%	66.63%	78.42%
	CoMoFoD	200	87	113	69	33	34.50%	61.06%	80.49%
BusterNet [WAAN18]	CASIA2	1313	557	756	146	146	11.12%	19.31%	76.84%
	CoMoFoD	200	78	122	33	33	16.50%	27.05%	80.49%
Multi-branch CNN [BPT19]	CASIA2	1313	489	824	536	-	40.82%	65.05%	-
	CoMoFoD	200	87	113	62	-	31.00%	54.87%	-

a minimum distance of 160 pixels. The results of this tuned DF-CMFD can be found in the last column of Table 5.1, which has the best F1-score for all the three evaluation protocols. In particular, the scores for "Pixel Level Evaluation Protocol B" are largely improved, reflecting a more accurate localization. More accurate masks imply that the ground-truth boundaries are more surely included, which is beneficial for our method. It is possible to obtain even better score by cross-validation on the hyper-parameters. However, to follow a realistic operational scenario, we use the empirical tuning and set the tuned DF-CMFD as our first-stage copy-move detector in the following experiments.

5.2.2.3 Results of source and target disentangling

The results of the BusterNet method for discerning source and target on both CASIA2 and CoMoFoD datasets are presented in the second block of Table 5.2. The "Opt-In" subset from [WAAN18] corresponds here to a subset we call "Favorable Subset" (Fav. Sub.). It has 190 images for CASIA2 and 41 for CoMoFoD. On the CASIA2 dataset, BusterNet has an accuracy of 19.31% on the "Discernible" subset and an overall accuracy of 11.12% which are rather low. However, BusterNet has a high accuracy of 76.84% on the 190 favorable samples which is satisfactory.

The results of our GMM-based method with binary masks from the tuned DF-CMFD are presented in the first block of Table 5.2. We have competitive results on the "Favorable Subset" against BusterNet: an accuracy of 78.42% on CASIA2, and 80.49% on CoMoFoD. On the two datasets, the number of images with correct source/target disentanglement by our method is more than three times that of BusterNet (618 vs. 179 images). The "Discernible" subset is a little bigger than the one of BusterNet and the performances are largely improved.

We also provide results for the deep-learning-based method from the recent arXiv pre-print [BPT19]. We use the code from the authors [3]. For a fair comparison, we feed the CNN of [BPT19] with the same masks from the tuned DF-CMFD as used by our method. It should be favorable for the CNN to use a more accurate first-stage detector, both at image and pixel levels. Indeed, the multi-branch CNN method [BPT19] now correctly identifies source and target for 536 images on CASIA2, compared to 357 images with the default parameters as reported in the original arXiv pre-print. This method does not provide a "Favorable Subset" to compare on. For our method and the method of [BPT19], there is no significant difference in the number of correct images on the two datasets, *i.e.,* 549 vs. 536 and 69 vs. 62, our method being slightly better. The performance is lower for both methods on CoMoFoD, mainly due to the higher difficulty of this dataset. In CoMoFoD, contrary to CASIA2, there is no manipulation, *e.g.,* scaling, on the target area.

At last, with curiosity, we did some post-experimental analysis of our GMM-based method and found that we are able to reach 100% of accuracy for source/target disentangling when the predicted mask from DF-CMFD shares at least 50% of duplicated areas with the ground-truth. We only mention this result as illustration of the importance of the localization accuracy of the first-stage detector. It is not a realistic scenario as it requires access to ground-truth masks. By contrast, although some images are in theory "Discernible", the source/target disentangling is very difficult on them because of poorly identified duplicated areas and areas of false alarms in the mask produced by the first-stage detector. This results in a success rate lower than 50% on such difficult images for both our method and [BPT19], *e.g.,* mistakenly attributing reversed labels or two same labels to ground-truth source and target areas. More efforts shall be devoted to the study of these difficult cases.

5.3 Variation of our method with PixelCNN [OKK16]

As explained earlier, GMMs are very efficient to model natural images and compute likelihood. To complete this study, we have explored another family of models. GMMs are parametric models and the choice of the parametrization is critical for performance. To get rid of the parametrization, one would seek to remove an arbitrary prior choice that could actually be dealt with inside the algorithm, therefore improving flexibility. To compute likelihood, it is also possible to use auto-regressive models. In the family of auto-regressive models, we focus on PixelCNN [OKK16]; [Sal+17] models. They are CNNs based on deep-learning. In this section, we will first present these models and then the variation of our method based on

3. https://github.com/andreacos/MultiBranch_CNNCopyMove_Disambiguation (Last accessed: Sept. 2020).

it.

5.3.1 PixelCNN model

The choice of this CNN model has been motivated by [UVL18] which suggests that it forms in itself a strong prior on the local structure of natural images: "the network does contain knowledge about the *low-level structure* of natural images". It is especially interesting for our problem as we seek for inconsistencies in low-level features of images. In addition, it would allow us to explore deep-learning methods and the associated framework and software particularities for the specific forensic problem of source and target disentangling in copy-move.

PixelCNN [OKK16] aims to model $p(\mathbf{x})$ where $\mathbf{x} = (x_0, .., x_n)$ is a vector built from a 2D matrix. In our case, it is a vector of $n = 64$ pixels corresponding to 8×8 patches. Thanks to the **chain rule** for joint distribution, we can write $p(\mathbf{x}) = p(x_0) \prod_{i=1}^{n} p(x_i|x_{i-1}, x_{i-2}, .., x_0)$. This is an auto-regressive model as the probability of a pixel is expressed conditionally to the previous neighbor. The pixels are predicted sequentially following the line from left to right in the image. This is arbitrary and other orderings could be consider as in [Ger+15]. According to our experiments, orderings are quite equivalent for performance, therefore we keep the same ordering as the one used in [OKK16].

In practice, to compute the log-likelihood of the vector, the auto-regressive modelling implies that each pixel's probability $p(x_i|x_{i-1}, x_{i-2}, .., x_0)$ should be computed sequentially. This is achieved thanks to the so-called masked convolution, as illustrated in Figure 5.4. The weights for blue pixels are equal to 1 while the weights for the masked white pixels are 0.

PixelCNN also uses **residual blocks**. A residual block is illustrated in Figure 5.5. It contains an identity mapping from the input to the output of the block, in addition to the traditional convolutional layers. The advantages of residual blocks are:

— Stronger gradients to back-propagate, therefore facilitating the training of deeper architectures or leading to faster learning;

— More complex information is propagated forward. In fact, it is almost not possible for convolutions with non-linearity such ReLU to learn an identity mapping between the input and the output. With a residual block, it is easily achieved and therefore it allows more complex information to be learned.

The first version of PixelCNN was pretty heavy and hard to make it converge. Some tricks and improvements were proposed in [Sal+17] and the improved network was named PixelCNN++. The main improvements are at the output layers. For each pixel in the image

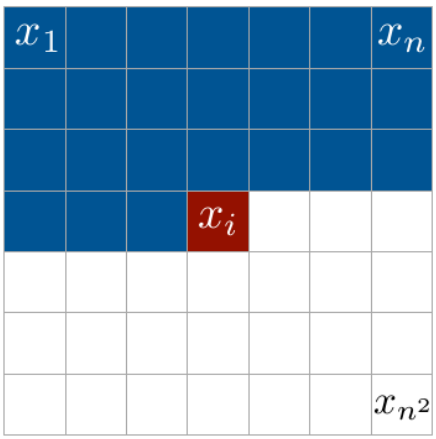


Figure 5.4 – To generate pixel x_i, all white pixels are masked and only blue pixels are allowed. This Figure taken from [OKK16].

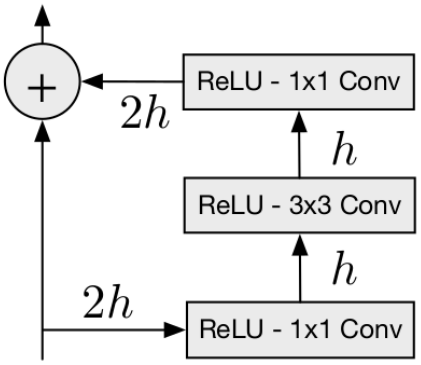


Figure 5.5 – Residual block of PixelNN. Figure taken from [OKK16].

patch (*e.g.* an 8×8 patch of 8-bit pixels), PixelCNN has 255 neurons on the output layer to predict the category of the pixel, *i.e.,* the integer pixel value between 0 and 255. It is very large. With PixelCNN++, a discretized logistic mixture is learned instead as the output. This mixture has $[0, 255]$ as support, so instead of 255 neurons per pixel they suggest to use 10 components in the mixture. It is also more flexible.

5.3.2 The method

The method is very similar to the one with the GMM. But we use here a PixelCNN++ model to estimate the log-likelihood. PixelCNN++ is first pre-trained on all the pristine zones of the available images. Then a fine-tuning is performed for each image to make the

model specific to the image, just as one GMM is trained per image. It was not possible to train from scratch one PixelCNN++ model per image as with the GMMs, because more samples are needed for the training of such a large CNN. For instance and on average, only about 80000 training sample are available per image with the CASIA2 database. It makes a pre-training necessary. Anyway, we have tested to train from scratch one PixelCNN++ per image but we were not able to make models converge.

5.3.3 Experiments

The metrics and datasets used in this subsection are the same as those in Section 5.2.2.1. The "Favorable Subset" is obtained with a ranking, just as with the GMM-based method. We use a Pytorch implementation of PixelCNN++ [Sal+17] than can be found at https://github.com/pclucas14/pixel-cnn-pp.

The results and a comparison with the one obtained by the GMM-based method are detailed in Table 5.3. We use the same masks obtained with the tuned version of DF-CMFD as in the previous Section 5.2.2.2. The overall performance of the method based on PixelCNN++ is slightly lower than the one obtained with the GMM. For the CoMoFoD dataset, the accuracy for the two methods is closer for the "Discernible" and "Favorable subset" than on CASIA2. On CoMoFoD, the performances of PixelCNN++ are also slightly better than on CASIA2 for the "Discernible" subset. It is promising, as we know that copy-move areas are never manipulated in CoMoFoD, contrary to CASIA2, so statistical deviation is smaller. We believe that better results could be achieved with more efforts on the design of the architecture. We have conducted these experiments on PixelCNN++ at the very end of this thesis work and we ran short of time to explore architecture alterations. Beside that, the idea was more like a proof of concept on the interest of PixelCNN++ for the detection of statistical deviation than the proper development of a method with the best results achievable. Here, we were more interested in identifying a future research line than maximizing pure performance.

5.4 Comparisons of the different approaches

In this section we study the differences between the two other existing methods [WAAN18]; [BPT19] and ours. We also provide a comparison of our two approaches, respectively based on GMM and PixelCNN++. We look for explanations to the different experimental performances described earlier in order to identify promising leads.

Table 5.3 – Source/target disentangling performances of GMM-based method and its variant with PixelCNN++. "Indisc." means the "Indiscernible" subset, "Disc." the "Discernible" subset, "Corr." are images with correct classification of source and target, "Fav. Sub." is the "Favorable Subset". Please refer to Section 5.2.2.1 for details of the metrics used in the table.

		Number of images					Accuracy		
	Dataset	Total	Indisc.	Disc.	Corr.	Corr. in Fav. Sub.	Overall	Disc.	Fav. Sub.
PixelCNN++ and tuned DF-CMFD	CASIA2	1313	489	824	465	127	35.41%	56.43%	66.84%
	CoMoFoD	200	87	113	67	30	33.50%	59.29%	73.17%
GMM and tuned DF-CMFD	CASIA2	1313	489	824	549	149	41.81%	66.63%	78.42%
	CoMoFoD	200	87	113	69	33	34.50%	61.06%	80.49%

5.4.1 Comparisons with [WAAN18] and [BPT19]

We were able to improve largely, when compared to BusterNet, the overall accuracy for source and target disentangling on both the CASIA2 and CoMoFoD datasets. One possible explanation is that contrary to BusterNet that relies on manipulation detection to locate and distinguish between source and target, in our method we assume that the boundary of the target area would expose statistical deviation. This deviation would be even larger if the interior of the target area is manipulated. A major limitation of BusterNet is that it tends to attribute the same label for both source and target areas, *i.e.,* no decision, while our method produces a decision on more images. In fact, BusterNet produces a decision only for 14.5% of the CASIA2 images. At last, the BusterNet method [WAAN18] does not enforce pixel correspondence in the source and target. DF-CMFD imposes such a prior which is beneficial for the method based on it, ours and [BPT19]. Beside that, BusterNet uses 256×256 images as input due to memory limitation, therefore test images are resized prior to being fed to the network. We can consider that this resizing acts as a post-processing, which would cover some fingerprints left by the manipulation on the target area or abnormal transitions between target and pristine areas. In contrast, our method is able to process full-sized images to discriminate source and target.

Comparable results on the CASIA2 dataset of our GMM-based method and [BPT19] could be explained by a similarity of the two methods: GMM-based or CNN-based approach driven by information from the copy-moved zones and boundaries. It seems, according to the results on the CoMoFoD dataset, that our method based on GMM is slightly better when no manipulation has been added to target area, *i.e.,* when statistical deviation is smaller. The method in [BPT19] needs a large synthetic dataset for training the network, while our GMMs are specific for each image. This probably allows us to capture more subtle differences. This single image setting also makes the development lighter as the training process is simpler and requires less resources. In contrast, CNN-based methods heavily rely on a large amount of

training data. For instance, the authors of BusterNet [WAAN18] have produced 100000 synthetic samples for the training of their network. Two additional external image manipulation datasets were also used for training. The network of [BPT19] has been trained on a synthetic dataset of 900000 samples. We can observe that contrary to the two deep-learning-based methods, no labels are used in the training of our method with a statistical machine learning tool of GMM. Loosely speaking, our approach could be considered like an unsupervised method working on a single image. This provides additional flexibility compared to the two CNN-based methods. Usually, classical machine learning tools provide lower performance than recent deep-learning approaches. Here we do not observe this trend and we explain it mainly by the adaptability of our method to the individual given image. This interesting point is worth further studying.

5.4.2 Comparison between GMM and PixelCNN++

GMM and PixelCNN++ are two different but related types of models. They are both learned with likelihood maximization. They both learn dependence between pixels. It is reflected by the covariance matrices in the GMM and by both the auto-regressive form and the convolutionnal kernels for PixelCNN++. Patches are vectorized for the GMM but covariance matrices capture the spatial dependencies of the patch. For PixelCNN++, as it is a CNN model, the patch is kept as it is when fed to the neural network. Therefore, the CNN takes directly advantage of the spatial dependencies inside the patch. The ordering for sequential conditioning of probability is a bias but it is done similarly to vectorization: from the top left to the bottom right. PixelCNN++ allows the use of color images while it would require three separate models from GMM to do so. The dependence between color channels should provide additional hints.

The number of parameters is slightly lower with GMM compared to PixelCNN++, 51225 compared to 74970. Yet, the number of hyper-parameters is smaller with GMM, as there are only two: the number of components and the type of covariance matrix. With PixelCNN++, more hyper-parameters are involved: the optimizer and its parameters (at least the learning rate and decay), the batch size, *etc.*, and last but not least the architecture has to be designed. It is done through a lot of experiments and consumes a large amount of computing power.

5.5 Summary

After a study of existing copy-move detectors, we propose a simple method to discriminate source and target areas in copy-move forgeries. Our approach acts as a second-stage detector and takes as input the binary mask produced by a first-stage copy-move detector. The basic idea is to measure and compare the statistical deviation of duplicated areas by using a Gaussian Mixture Model trained on identified pristine patches. We show that our method outperforms the only other published detector capable of such a disentanglement, BusterNet [WAAN18], on two different datasets (CASIA2 and CoMoFoD). Another advantage of our method is the possibility to rank predictions among a dataset to extract images with the most surely distinguished source and target areas. As discussed in Section 5.4, we consider that bringing additional flexibility and adaptability with the single image framework compared to the training of deep-learning methods on large synthetic datasets is a key factor for performance. Usually, deep-learning methods are able to achieve (much) better results than classical machine learning tools, such as GMMs, but this is not the case here. It is thus a promising line of research to find solutions to make deep-learning methods more flexible and adaptive.

We have then proposed a variation of this approach with the PixelCNN++ model instead of GMM for local statistics modeling. PixelCNN++ performs slight worse but gives rather similar performance when compared to GMM. It would be possible to fuse results from the two models to improve the accuracy. Beside that, it has been experimentally shown that PixelCNN++ is also a good candidate for local modelling of natural images. With more time and experience with CNN, it seems possible to improve the modelling capability of PixelCNN++. Especially, it would be promising to design an architecture specifically for small patches of 8×8 pixels. Here, the architecture is designed for patches of at least 32×32. It would also be interesting to carry out some tests with bigger patches but this would require more computing resources.

As future work, other statistical models and histogram distance measures could be considered. Another interesting working direction would be an approach without statistical modeling like in GMM, but directly with the pixel values. This would then probably require more advanced distance measures such as Wasserstein or *Maximum Mean Discrepancy* (MMD). A metric could also be learned with a neural network. Siamese networks seem especially promising, such as those used in [BPT19]. We plan to test our method combined with more first-stage detectors to study their impact. There is also room for improvement in the post-processing of the first-stage detector by using useful information provided by our method, *e.g.,* the ranking scores. It could help to discard the false positives of the first-stage detector.

6 | Conclusion and Perspectives

With the prominent importance and role of images in our society, it is a necessity to design tools capable of assessing the integrity of images. At the same time, the variety of scenes, sources and types of tampering makes the diversity of these images enormous. In this context, it seems unrealistic that a detector would be able to perform effectively and with the same performance on all images. Flexibility and adaptability to the data seem to us a crucial point that have guided all the work of this thesis. The development of high performance models is above all important but in all our methods a special care about flexibility and adaptability has been taken.

6.1 General conclusion

In Chapter 2, we have first introduced how the digital images are captured and we have described the image forensics field. Then we have introduced the different problems that are tackled in this thesis: manipulations detection and falsifications detection and localization. A state-of-the-art of existing methods is provided. Gaussian Mixture Models and neural networks are the two main tools used in the methods developed in the course of this thesis. These tools relate to a broader category called *Machine learning*. Therefore, a theoretical background for these methods has been outlined.

In Chapter 4, we have presented an original problem of image forensics. We consider image manipulation detection in the case of pre-processing, more precisely when a resizing of the full-sized image is performed prior to the addition of a manipulation. Images are commonly down-sized for storage or display purposes. So, it is quite realistic to assume that someone would retrieve an already down-sized image to then add a manipulation. In such a scenario, we experimentally show that the performances of state-of-the-art detectors drop. An illustration of the drops in accuracy can be found in Figure 6.1, for the CNN-based classifier from [BS16]. We have performed statistical tests to formally expose statistical deviations of

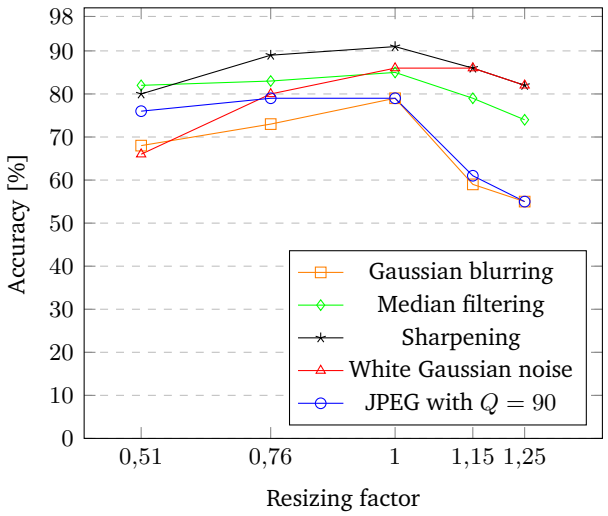


Figure 6.1 – Testing accuracy (in $\%$) of CNN-based method [BS16] for detection of 5 different types of manipulation with pre-resizing. The X-axis is the resizing factor. A resizing factor of 1 means no resizing and is considered as the baseline performance. The resizing algorithm considered here is the harder bi-cubic interpolation.

the pre-resized images. Then, we have proposed three different methods for adaption on two detectors. The objective is to recover some performance through adaptation of the model to the new data. The first method is quite classical as it is the fine-tuning of a neural network. The two others, designed for a classification pipeline that uses Gaussian Mixture Models, are original. These two methods are used in two different settings: if no labels are known on pre-resized data (unsupervised) or if only a few labels are known (weakly supervised). The first approach performs geometrical transformations on covariances matrices of the GMMs to adapt them to the new empirical covariance of the data. For the second method, the GMMs are used in the pipeline as features extractor. The weights of the GMMs are updated to better fit empirical covariance of the data and the classifier is fine-tuned with the few labeled samples available on the target. The GRAFT procedure is able to recover up to $+32\%$ for the detection of Gaussian filtering with a pre-resizing of $\times 1.25$. For the weakly supervised method, it is up to $+18\%$ for `JPEG` detection and pre-resizing of $\times 1.25$.

Finally, in Chapter 5 we have shifted our focus on falsification detection. In the broad range of existing falsifications, we have concentrated on copy-move detection. It is, along with splicing and inpainting, one of the most common and popular tampering method. Several very powerful copy-move detectors are already available. Although most of the existing detectors identify indifferently the source and the target area of the copy move. From an operational point of view, it is more valuable to identify only the tampered area and therefore to

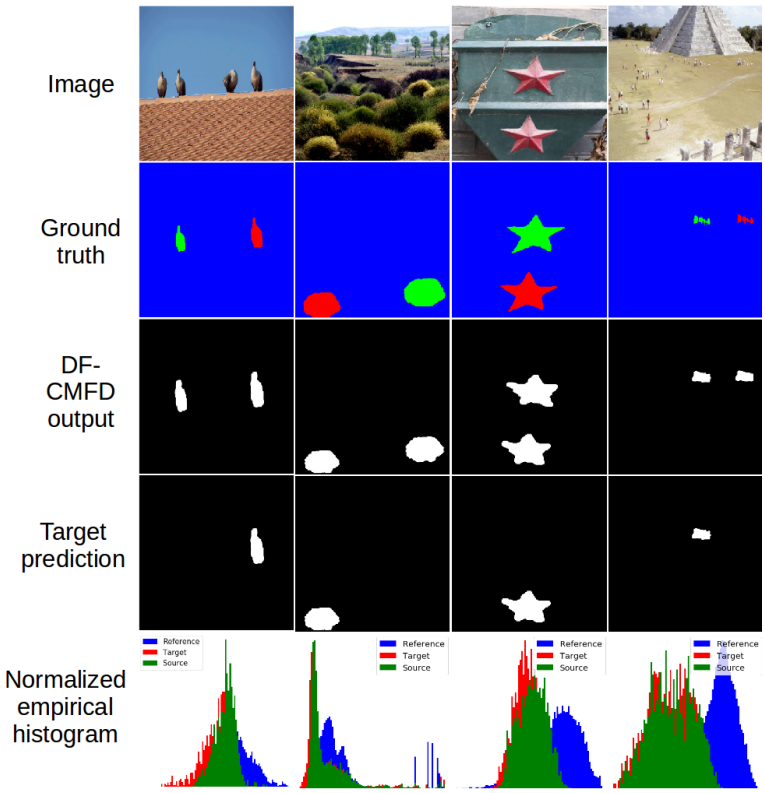


Figure 6.2 – Example results of our GMM-based method with DF-CMFD [CPV15a] as the first-stage copy-move detector. The red color in ground-truth masks indicates the target area, the green color indicates the source one, and the blue color indicates the pristine part. The example in the last column is a miss-classification.

disentangle the source and target areas. For instance, if an image is used as evidence during a trial and an individual is copy-moved, it is important to distinguish the two zones to be able to locate the character in the scene. Only one published method and a pre-print have proposed solutions to this end. We have introduced a new flexible and adaptive method as it is specific to each image and does not require a large database for training. Our method shows comparable or slightly better results with more versatility, when compared to the existing method. An illustration of the typical output of an existing detector and what our method proposes and manages to do are depicted in Figure 6.2.

In addition to the above work, we also present briefly in Appendix A, the methods used by our team during the different rounds of the DEFALS challenge. The DEFALS challenge has been organized by ANR-DGA who have funded this work. It was a competition of image falsifications detection on two rounds, between four French teams and on an original dataset provided by ANR-DGA. To this end, we have engineered methods from the literature. This contest was a unique opportunity of concrete experimentation that also has greatly influenced

the theoretical work of this thesis. The emphasis on flexibility and adaptability was largely inspired by our experience during the contest.

6.2 Perspectives and future work

We list here some perspectives based on the experience of this thesis and the problems encountered. Some perspectives are *short-term*, direct evolution and variations of the proposed and existing methods. They are directly related to the problems tackled in this thesis. Some others are rather *long-term* perspectives as they concern promising lines of work and problems not considered in this work.

Short-term perspectives

Regarding our research on image manipulation detection, the original GRAFT procedure could be improved with the use of a few labels if they are available, like in the weakly-supervised method. The use of the labels would allow the computation of more accurate covariance matrices for the target data. In the current version of GRAFT, to find optimal interpolation parameters, the log-likelihood is maximized on the target data. With few labels, the accuracy score could be directly optimized. However, some studies should be carried out to evaluate how many samples are required to bring an improvement compared to the unsupervised adaptation framework. Finally, it could also be interesting to combine results of unsupervised and weakly-supervised GRAFT algorithms with a factor that depends on the number of available labels on the target. It would bring even more flexibility to the method regarding target data. On another side, we have mentioned feature adaptation in Section 4.4.1. Yet, we did not succeed to adapt directly the features instead of the model as in the two proposed methods. Recent advanced in the field of Optimal Transport could be investigated to test new methods. For example, very recently an approach that combines feature and sample transportation [Red+20] has been proposed.

Beside that, for copy-move detection more experiments with other first-stage copy-move detectors should be conducted. It would be interesting to quantify the effects of the first-stage performance on the second stage. Intuitively, we could expect that with less accurate copy-move masks the performance of the second stage would decrease. Boundaries would not be identified precisely, which may lead to incorrect log-likelihood comparisons. An intermediate step of areas matching in copy-move detector output would also probably provide better results. Some copy-move detectors already provide such information, like [CPV15a], although we do not use it. A general procedure, independent of the first-stage copy-move detector,

could be designed and developed as well.

Finally, it would be interesting to apply our work and source/target disentangling for copy-move to the second-round dataset of DEFALS to measure the improvement on the F1-score. We also had noticed that double JPEG compression detectors give interesting results for a limited set of images while for the majority it is random. A fusion strategy that identifies automatically this limited set would also provide an improvement on the F1-score.

Long-term perspectives

We recall that the main challenge to switch from the public benchmarks to image forensics in the wild is the difference in versatility of the data. Real-world tampered images are more diverse in terms of source, tampering type, pre- and post-processing. One promising line of research is probably to develop from scratch models that generalize well and not only models that perform well, like in [Coz+18]. Another promising line, which seems even more interesting to us, is to add flexibility and adaptability to models like we sought in this thesis. The fact that it allows classical methods to still be competitive with neural networks methods is for us a good indicator of the potential of the adaptive approach.

Considering this, it seems promising to pursue the work on auto-regressive deep models to provide them with additional flexibility. A common point of manipulations and most common falsifications is that they should expose some statistical deviations when compared to pristine areas, either on the full zone or on the boundaries. The strategy of deviation detection has already been applied to other problems than manipulation detection and copy-move disentangling, for instance splicing detection with [CPV15b]. Auto-regressive models are by design, as they are statistical models, tailored to be able to detect such deviations. More broadly, the development of more advanced adaptation techniques than fine-tuning for neural networks would be a great step toward the detection of highly realistic forgeries.

Finally, public benchmarks are really important to help the field progress. It appears that the DEFALS datasets are very realistic. It would be very beneficial to find solutions to make them publicly available. As for now, copyright legal issues on images are the main limitation. Beside easy access to high quality data for experiment, it would also be interesting to allow more collaborations and to share methods between teams that have taken part in the DEFALS challenge. We probably all have a lot of intuitions and knowledge to share that could lead to new ideas when combined together.

A | The DEFALS Challenge

Contents

In this Appendix, we present succinctly the methods implemented and what we have learned on image forensics during the DEFALS challenge.

A.1 Preliminary round

For this first round of binary classification between original and falsified images, our best score has been obtained by matching test images against a set of previously released training images. The most efficient technique to perform the matching was based on *hashing*. Except for the specific identity function used *e.g.*, in Java, hash functions map data of arbitrary size (usually big) to a fixed-size (usually small) binary digest, which make hash functions injective. Typically (and as we did), we used the plain image files bytes as the contents to be hashed. In our case, we used the `sha256` hash function that produce 256-bit long digest with an extremely low probability of collision, thus these hash codes actually make for unique identifier of the images. We then simply looked for matches of the hash codes. An extra check was also performed on image size (in bytes) to ensure correspondence (although image sizes were identical, the `EXIF` metadata had different sizes).

We also have quickly explored a matching using features extracted with classical pre-trained CNN, antoher classical approach for this kind of problem. Yet, the feature extraction process was longer and the features had bigger size, which would induce a longer computing time for matching. As said above, we had also noticed that `EXIF` metadata had not been erased. Thus, the `EXIF` tags of falsified images contained a string describing the software used to perform tampering. Although this matching was good enough to obtain a perfect identification of falsified images, we did not use it and we just notified the organizers of the challenge about it.

As this round has occurred during the very first months of the thesis, it was also an opportunity to test several methods and their implementations of the literature. The findings were that most of the classical methods performed poorly on such realistic and high quality dataset without proper adaptation and modifications. The best results, though limited, were obtained with steganalysis features (SRM and SPAM) and a regular classifier.

A.2 First round

As described in Section 1.2, the objective was to perform binary classification of falsified images. Beside the method based on metadata and presented in Section 2.2.3.2, we have

developed two methods and a fusion scheme.

The first method is based on classical steganalysis features: SRM color [GFC14]. They are extracted from 256×256 patches randomly cropped in the images. As explained in Section 2.2.3.2, the color dynamics and range of falsified and original images are different. Therefore, a simple linear classifier on SRM features is able to distinguish with good performance falsified images and original images. During the testing phase, several patches are extracted from each image and a majority voting is carried out. A linear classifier also allows us to rank the images according to the distance to the hyper-plan of classification. To avoid too many false positives that would penalize our score, we set a threshold in this ranking to only keep the first few images which are the more suspicious ones.
We also have developed a slight variation of this method with one-level SRM features on the Cr channel of YCrCb space. The intuition here is that the difference in color dynamics should be more visible in the YCrCb color space. In fact, the difference was big enough in the RGB color space so that performances are comparable. We also have tested other, more elaborate classifiers such as XGBoost [CG16] or RandomForest [Bre01], still without significant improvement. SRM features are quite long to extract so we took advantage of the computing cluster of Lille University and GRICAD in Grenoble to parallelize the computation.

The second method was deep-learning based. We use a network developed by two of the team member [CW19] for image manipulations detection. The change in color dynamics is a manipulation more than a falsification so it makes sense to use this network. Similarly to the feature-based method, inputs are 256×256 patches randomly cropped from the images and a majority voting is performed during the test phase. The last layer is a typical softmax layer. It produces a score that is usually interpreted as the probability of belonging to the class (in our case, the class of manipulated patches). We use this score to construct a ranking just as with the other method and set a threshold to obtain a set of the most suspicious images.

To merge the three rankings, the two SRM and the deep learning-based, we have firstly used a basic method. For each method, we extract the 200 most suspicious images and we take the 80 images in the intersection. The number 80 was an estimation of the total number of falsified images in the test set. Both methods achieved very similar results so the most suspicious images for both methods were also close. An intersection with the chroma subsampled images was also performed, as explained in Section 2.2.3.2. We also have explored advanced methods for ranking fusion. Most notably, we have implemented a *Borda count*. In Borda count for each ranking, samples are given a number of points that corresponds to the number or samples that are below in the ranking. Thus for the three rankings (the two SRM and the deep learning-based method), each sample has a number of point corresponding to its position in the specific ranking. Then, points from each ranking are added to obtain the

Table A.1 – Left: we have four samples A, B, C, D and 100 rankings (votes from 100 voters). A is first in 51 rankings and last in all the others. C is first in only 5 rankings and second in all the others. Right: Score for C is $5 \times 3 + 95 \times 2 = 205$. A have a score of $51 \times 3 + 49 \times 0 = 153$. Thus C is first and A is second, while A have been place first by a majority (51 out of 100 rankings). Example from: https://en.wikipedia.org/wiki/Borda_count (last accessed: Sept. 2020).

Rank	51 rankings	5 rankings	23 rankings	21 rankings
1st	**A**	**C**	B	D
2nd	**C**	B	**C**	**C**
3rd	B	D	D	B
4th	D	**A**	**A**	**A**

Sample	Borda score
C	**205**
A	153
B	151
D	91

new ranking. The biggest counterpart with Borda count is that a sample in first place for a majority of rankings is not necessarily the first one in the fused ranking. An example is depicted in Table A.1. However, as we only have 3 rankings in our setting for the DEFALS challenge, it is not expected to happen. We also have tested the *Condorcet method*, which is based on the intuition that the best candidate in an election is the one that would win the most showdowns against the others. For this method, we therefore construct all pairing of samples. Each sample obtains a number of points corresponding to the number of pairings where it appears higher in rankings than the other sample in the pairing. Thus, the sample at the top of the merged ranking is the one that "won" most showdowns against the other samples according to the three rankings. Both of these methods only provide a marginal improvement. Rankings are too similar, so the output of a simple intersection is not really different from the outputs of advanced methods.

A.3 Second round

This round had two objectives, and two scores: *(i)* identification of falsified images, and *(ii)* tampering localization. This time, we witnessed no `EXIF` leaks and no chroma subsampling or obvious color dynamics differences in the dataset.

A.3.1 Identification

The methods used during the first stage, feature-based and deep-learning based, gave poor results as there are no significant statistical difference in the distributions of falsified

images compared to the original ones: falsifications were only local. So we had to devise new methods again. The identification was especially hard because of the high quality of the falsifications. Our original strategy was to first identify falsified images and then perform a second stage of localization. Because of these difficulties, we directly have aimed for a detector capable of localization. As mentioned in Section 2.2.3.2, some image sizes were found to be specific to falsified images, so we were able to identify majority of them in this way.

A.3.2 Localization

Our best results for localization were achieved using a vanilla (with default parameters) version of [CPV15a]. This detector is able to identify and localize most of the copy-move and some exemplar-based inpaintings. For more details on the different types of inpaintings, see Section 2.1.3. The main limitation is that the detector identifies both areas of the copy-move while in this contest we were interested only in the tampered target zone. This lead us to work on disentangling source and target areas in copy-move. This work is presented in Chapter 5.

Beside the copy-move identification and localization, we also tried other detectors for detecting splicing and inpainting, both diffusion-based and exemplar-based. We have mainly worked on [Bap+19] and [WAN19]. They are two neural networks capable of general falsification detection. We have focused on these two because the reported results on public benchmarks look very impressive and authors proposed implementations as well as pre-trained versions of their methods. Both methods are based on CNNs and they are intended to work with full-sized images. The authors considered images of maximum size 512×512 in their experiments. Because of the very large size of DEFALS images and hardware memory limitations, it was not possible for us to work with the full-sized images. We tried to resize the images to 512×512 but this very strong post-processing made the detection very difficult. Therefore, we decided to work on crops instead of resized or full-sized images. The idea was to divide the images into 256×256 patches and then to merge decisions in order to gain full-sized localization. In this setting, the data used for training by the authors, mostly synthetic data, were too different from our high-definition patches. To tackle this issue, we tried to re-train and also fine-tune these networks.

We first encountered an issue of class imbalance. Most of the crops were from pristine parts of images, and in the crops containing falsified parts the percentage of falsifications would vary a lot. This made the training impossible because the pristine class was over-represented in the training set. Thus, we only kept 20% of fully pristine patches. For the

falsified patches, we only used patches containing between 20% and 80% of falsified content, to ensure that significant fingerprints of falsification would be observed. All these proportions have been set empirically. We spent a lot of time experimenting the construction of a training dataset that would be representative of our data and that would also permit the networks to learn something on and to converge. This dataset construction is the crucial part that allowed us to make networks converge and achieve good results. The usual classification loss function in neural networks is the cross-entropy loss which requires balanced label distributions. It is definitely not the case in our training data, so we did some experimentation with weighted cross-entropy, Dice loss and especially focal loss [Lin+17]. Weighted cross-entropy loss adds a weight to one of the two classes to reflect the imbalance of classes. In our case, we estimate that in the image the falsified part represents between 10% and 20% of the pixels. The Dice loss is a score based on the Dice coefficient which measures the similarity of two ensembles, in our case the ensembles of pixels between predicted mask and ground truth. Along with the Jaccard index, the Dice loss is commonly used for object detection in images as it counterbalances partially the imbalance of classes. The focal loss adds a term to the classical cross-entropy loss in order to reduce the relative loss for well-classified examples, *i.e.* examples correctly classified and with high certainty (probability from softmax close to 1). The intuition is that there is more to learn from hard examples than easily classified ones. We have first performed experimentation on these losses separately. Then, to benefit from the better of all these losses, we have combined them to form a new loss. It allows us to obtain good results of localization at the patch level. The results are especially good with splicing as this tampering exposes edges between the two zones. Because of the nature of convolution (a filter), CNNs are particularly powerful to detect edges. However, the results are very poor at image level. When merging all the patches prediction, the number of false positives made the localization very fuzzy. Although we invested a lot of time and resources in this research line, we were not able to use it for the challenge.

Finally, we also have explored some methods which detect re-sampling traces. We had the intuition that copy-move, splicing or exemplar-based inpainting would usually imply re-scaling or rotation of the tampered area to look more natural. Re-scaling or rotation are basically re-sampling and this would be visible in for example the Fourier domain with the appearance of some periodic peaks. Probably these fingerprints in the Fourier domain are covered by the latter JPEG compression performed after the falsification and before the distribution of images.

To summarize, both the data and the goals of this round made it especially hard. It is reflected by the lower scores achieved by all the teams compared to the previous rounds. It also has confirmed to us that it is not straightforward to adapt efficient detectors from the

literature to new and realistic data. It requires specific work and studies. With more adaptable and flexible models, it would be definitely much easier.

A.4 Summary

This challenge was a pretty unique opportunity to work on very high-quality datasets. It is very representative of what it requires to move from synthetic benchmarks to image forensics in the wild. It also has allowed us to test in practice, through implementations, a wide range of methods from the literature. Concrete experimentations of the challenge have enabled us to explore extensively frameworks and hardware considerations around deep-learning. It was truly complementary with the other parts of the thesis, which is more theoretic and focused on classical approaches.

The last round has exposed how the localization of high-quality falsifications in images of big size is a difficult problem. It is certainly a major challenge that still needs to be specifically addressed in the image forensics field. One crucial issue is the availability of the datasets that is dependent on legal issues, but it would be very profitable for the community to share the datasets of the three DEFALS challenge rounds.

One regret that we have with this contest is that the collaboration between the teams has been minimal. After each round, debriefing meetings were mainly about the technical conditions and the following rounds of the contest. The methods developed by the different teams have not been shared or explained in details. That is probably because debriefing meetings were intended for team leaders. It would have been profitable to also have a debriefing for the team members that have worked in more details on devising the methods. Complementary with the competition, a collaboration stage between teams, not only with the industrial partner of each team, would certainly foster the development of new ideas.

B | The softwares

All the experiments and practical experimentation of this thesis could not have been implemented without the help of open software and libraries. We cite them here as acknowledgements but also as these tools often rely on number of citations for budget.

Firstly, we have used Python [RD09] as programming language. All the libraries from the ScyPy [Vir+20] ecosystem have been of a great help: scientific computation with NumPy [WCV11] and the plotting library Matplotlib [Hun07]. All implementations of the classical machine learning algorithms that we used are from Scikit-Learn [Ped+11]. For deep learning, we used mainly PyTorch [Pas+19] and Tensorflow [Aba+16]. For the computer vision part, we used mainly OpenCV [Bra00] and Pillow [Cla15].

Bibliography

[Aba+16] M. Abadi et al. “Tensorflow: A system for large-scale machine learning”. In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. 2016, pp. 265–283.

[AW19] A. Azulay and Y. Weiss. “Why do deep convolutional networks generalize so poorly to small image transformations?” In: *Journal of Machine Learning Research* 20.184 (2019), pp. 1–25.

[Bap+19] J. H. Bappy et al. “Hybrid LSTM and encoder–decoder architecture for detection of image forgeries”. In: *IEEE Transactions on Image Processing* 28.7 (2019), pp. 3286–3300.

[Bar+09] C. Barnes et al. “PatchMatch: A randomized correspondence algorithm for structural image editing”. In: *ACM Transactions on Graphics* 28.3 (2009), 24:1–24:10.

[BD+06] S. Ben-David et al. “Analysis of representations for domain adaptation”. In: *Proceedings of the International Conference on Neural Information Processing Systems*. 2006, pp. 137–144.

[Bis06] C. M. Bishop. *Pattern recognition and machine learning (information science and statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[BPT19] M. Barni, Q. Phan, and B. Tondi. “Copy move source-target disambiguation through multi-branch CNNs”. In: *arXiv CoRR* (2019), pp. 1–13.

[Bra00] G. Bradski. “The OpenCV library”. In: *Dr. Dobb’s Journal of Software Tools* (2000).

[Bre01] L. Breiman. “Random forest”. In: *Machine Learning* 45 (Oct. 2001), pp. 5–32.

[BS16] B. Bayar and M. C. Stamm. “A deep learning approach to universal image manipulation detection using a new convolutional layer”. In: *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*. IHMMSec ’16. ACM, 2016, pp. 5–10.

[Cao+11] G. Cao et al. “Unsharp masking sharpening detection via overshoot artifacts analysis”. In: *IEEE Signal Processing Letters* 18.10 (Oct. 2011), pp. 603–606.

[Cao+14] G. Cao et al. “Contrast enhancement-based forensics in digital images”. In: *IEEE Transactions on Information Forensics and Security* 9.3 (Mar. 2014), pp. 515–525.

[CG16] T. Chen and C. Guestrin. “XGBoost: A scalable tree boosting system”. In: *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining*. 2016, pp. 785–794.

[CGB19] R. Cogranne, Q. Giboulot, and P. Bas. “The ALASKA steganalysis challenge: A first step towards steganalysis”. In: *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security*. IHMMSec’19. New York, NY, USA: Association for Computing Machinery, 2019, 125–137.

[CGV15] D. Cozzolino, D. Gragnaniello, and L. Verdoliva. “Image forgery localization through the fusion of camera-based, feature-based and pixel-based techniques”. In: *2014 IEEE International Conference on Image Processing, ICIP 2014* (Jan. 2015), pp. 5302–5306.

[Che+15] J. Chen et al. “Median filtering forensics based on convolutional neural networks”. In: *IEEE Signal Processing Letters* 22.11 (Nov. 2015), pp. 1849–1853.

[Chr+12] V. Christlein et al. “An evaluation of popular copy-move forgery detection approaches”. In: *IEEE Transactions on Information Forensics and Security* 7.6 (2012), pp. 1841–1854.

[Cla15] A. Clark. *Pillow (PIL fork) documentation*. 2015.

[Cou+17] N. Courty et al. “Optimal transport for domain adaptation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.9 (Sept. 2017), pp. 1853–1865.

[Coz+18] D. Cozzolino et al. “ForensicTransfer: Weakly-supervised domain adaptation for forgery detection”. In: *arXiv* (2018).

[CPV15a] D. Cozzolino, G. Poggi, and L. Verdoliva. “Efficient dense-field copy–move forgery detection”. In: *IEEE Transactions on Information Forensics and Security* 10.11 (2015), pp. 2284–2297.

[CPV15b] D. Cozzolino, G. Poggi, and L. Verdoliva. “Splicebuster: A new blind image splicing detector”. In: *IEEE International Workshop on Information Forensics and Security (WIFS)*. Nov. 2015, pp. 1–6.

[Cut13] M. Cuturi. “Sinkhorn distances: Lightspeed computation of optimal transport”. In: *Proceedings of the International Conference on Neural Information Processing Systems*. 2013, pp. 2292–2300.

[CV20] D. Cozzolino and L. Verdoliva. “Noiseprint: A CNN-based camera model fingerprint”. In: *IEEE Transactions on Information Forensics and Security* 15 (2020), pp. 144–159.

[CW19] I. Castillo Camacho and K. Wang. “A simple and effective initialization of CNN for forensics of image processing operations”. In: *ACM Information Hiding and Multimedia Security Workshop*. Paris, France, July 2019.

[Cyb89] G. Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of Control, Signals, and Systems (MCSS)* 2.4 (Dec. 1989), pp. 303–314.

[DAR17] DARPA. *Nimble challenge 2017 evaluation*. Dec. 2017.

[DWC19a] L. Darmet, K. Wang, and F. Cayre. “GRAFT : Adaptation non-supervisée au redimensionnement pour la détection de manipulation d’image”. In: *XXVIIème colloque GRETSI (GRETSI 2019)*. Lille, France, Aug. 2019.

[DWC19b] L. Darmet, K. Wang, and F. Cayre. “Weakly supervised adaptation to re-sizing for image manipulation detection on small patches”. In: *Proceedings of the International Workshop on Digital-forensics and Watermarking*. 2019.

[DWC20] L. Darmet, K. Wang, and F. Cayre. “GRAFT: Unsupervised adaptation to resizing for detection of image manipulation”. In: *IEEE Access* 8 (2020), pp. 55619–55632.

[DWT13] J. Dong, W. Wang, and T. Tan. “CASIA image tampering detection evaluation database”. In: *2013 IEEE China Summit and International Conference on Signal and Information Processing*. 2013, pp. 422–426.

[Fac12] Hacker Factor. *FotoForensics*. 2012.

[Fan+12] W. Fan et al. “3D lighting-based image forgery detection using shape-from-shading”. In: *European Signal Processing Conference*. Aug. 2012, pp. 1777–1781.

[FB08] T. Furon and P. Bas. “Broken arrows”. In: *EURASIP Journal on Information Security* 2008 (Oct. 2008).

[Fd03] Z. Fan and R. L. de Queiroz. “Identification of bitmap compression history: JPEG detection and quantizer estimation”. In: *IEEE Transactions on Image Processing* 12.2 (Feb. 2003), pp. 230–235.

[FK12] J. Fridrich and J. Kodovský. “Rich models for steganalysis of digital images”. In: *IEEE Transactions on Information Forensics and Security* 7.3 (June 2012), pp. 868–882.

[FSL03] J. Fridrich, D. Soukal, and J. Lukàš. “Detection of copy-move forgery in digital images”. In: *Proceedings of the Digital Forensic Research Workshop*. 2003, pp. 1–10.

[Fuk80] K. Fukushima. “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position”. In: *Biological Cybernetics* 36 (1980), pp. 193–202.

[FWC15] W. Fan, K. Wang, and F. Cayre. “General-purpose image forensics using patch likelihood under image statistical models”. In: *Proceedings of the IEEE International Workshop on Information Forensics and Security*. 2015, pp. 1–6.

[GB10] T. Gloe and R Böhme. “The ‘Dresden image database’ for benchmarking digital image forensics”. In: *Proceedings of the 25th Symposium On Applied Computing (ACM SAC 2010)*. Vol. 2. 2010, pp. 1585–1591.

[GBC16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT Press, 2016.

[Ger+15] M. Germain et al. “MADE: Masked autoencoder for distribution estimation”. In: *Proceedings of the 32nd International Conference on Machine Learning*. Vol. 37. Proceedings of Machine Learning Research. PMLR, July 2015, pp. 881–889.

[GFC14] M. Goljan, J. Fridrich, and R. Cogranne. “Rich model for steganalysis of color images”. In: *2014 IEEE International Workshop on Information Forensics and Security (WIFS)*. 2014, pp. 185–190.

[GKR13] Thomas Gloe, Matthias Kirchner, and Christian Riess. “How we learned to stop worrying about content and love the metadata”. In: *IFS-TC Image Forensics Challenge Special Session during WIFS* (2013).

[Gre+07] A. Gretton et al. “A kernel approach to comparing distributions”. In: *Proceedings of the National Conference on Artificial Intelligence*. 2007, pp. 1637–1641.

[Gre+08] A. Gretton et al. “A kernel statistical test of independence”. In: *Proceedings of the International Conference on Neural Information Processing Systems*. 2008, pp. 585–592.

[HTF01] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[Huh+18] M. Huh et al. “Fighting fake news: Image splice detection via learned self-consistency”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018.

[Hun07] J. D. Hunter. “Matplotlib: A 2D graphics environment”. In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95.

[IB89] H. F. Inman and E. L. Bradley Jr. “The overlapping coefficient as a measure of agreement between probability distributions and point estimation of the overlap of two normal densities”. In: *Communications in Statistics - Theory and Methods* 18 (1989), pp. 3851–3874.

[Kan+13] X. Kang et al. “Robust median filtering forensics using an autoregressive model”. In: *IEEE Transactions on Information Forensics and Security* 8.9 (Sept. 2013), pp. 1456–1468.

[KB15] D. P. Kingma and J. Ba. “Adam: A method for stochastic optimization”. In: *3rd International Conference on Learning Representations ICLR*. 2015.

[KJF11] E. Kee, M. Johnson, and H. Farid. “Digital Image Authentication From JPEG Headers”. In: *Information Forensics and Security, IEEE Transactions on* 6 (Oct. 2011), pp. 1066 –1075.

[KOF13] E. Kee, J. F. O’Brien, and H. Farid. “Exposing photo manipulation with inconsistent shadows”. In: *ACM Transactions on Graphics (ToG)* 32.3 (2013), pp. 1–12.

[KOF14] Eric Kee, James O’brien, and Hany Farid. “Exposing Photo Manipulation from Shading and Shadows”. In: *ACM Transactions on Graphics* 33 (Aug. 2014).

[KW14] D. P. Kingma and M. Welling. “Auto-encoding variational bayes”. In: *2nd International Conference on Learning Representations ICLR*. 2014.

[LeC+89] Y. LeCun et al. “Backpropagation applied to handwritten zip code recognition”. In: *Neural Computation* 1 (1989), pp. 541–551.

[Lee13] D. Lee. “Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks”. In: *ICML 2013 Workshop : Challenges in Representation Learning (WREPL)*. 2013.

[LG06] A. Langille and Minglun Gong. “An efficient match-based duplication detection algorithm”. In: *Proceedings of the Canadian Conference on Computer and Robot Vision*. 2006, 64:1–64:8.

[LHQ10] W. Luo, J. Huang, and G. Qiu. “JPEG error analysis and its applications to digital image forensics”. In: *IEEE Transactions on Information Forensics and Security* 5.3 (Sept. 2010), pp. 480–491.

[Li+18] H. Li et al. “Identification of various image operations using residual-based features”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 28.1 (Jan. 2018), pp. 31–45.

[Lin+14] T. Lin et al. “Microsoft COCO: Common objects in context”. In: *Computer Vision – ECCV 2014*. Cham: Springer International Publishing, 2014, pp. 740–755.

[Lin+17] T. Lin et al. “Focal loss for dense object detection”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2999–3007.

[Mar+20] F. Marra et al. “A full-image full-resolution end-to-end-trainable CNN framework for image forgery detection”. In: *IEEE Access* 8 (2020), pp. 133488–133502.

[MRF19] P. Mullan, C. Riess, and F. Freiling. “Forensic source identification using JPEG image headers: The case of smartphones”. In: *Digital Investigation* 28 (2019), S68 –S76.

[MS20] O. Mayer and M. C. Stamm. “Forensic similarity for digital images”. In: *IEEE Transactions on Information Forensics and Security* 15 (2020), pp. 1331–1346.

[MT20a] K. B. Meena and V. Tyagi. “A copy-move image forgery detection technique based on tetrolet transform”. In: *Journal of Information Security and Applications* 52 (2020), 102481:1–9.

[MT20b] K. B. Meena and V. Tyagi. “A hybrid copy-move image forgery detection technique based on Fourier-Mellin and scale invariant feature transforms”. In: *Multimedia Tools and Applications* 79 (2020), pp. 8197–8212.

[NC04] T. Ng and S. Chang. *A data set of authentic and spliced image blocks*. 2004.

[Net] *Netflix recommendations: Beyond the 5 stars*. Apr. 2012.

[OF12] J. F. O’Brien and H. Farid. “Exposing photo manipulation with inconsistent reflections”. In: *ACM Transactions on Graphics* 31.1 (Jan. 2012). Presented at SIGGRAPH 2012, 4:1–11.

[OKK16] A. V. Oord, N. Kalchbrenner, and K. Kavukcuoglu. “Pixel recurrent neural networks”. In: *Proceedings of The 33rd International Conference on Machine Learning*. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, June 2016, pp. 1747–1756.

[Pan+11] S. Pan et al. “Domain adaptation via transfer component analysis”. In: *IEEE Transactions on Neural Networks* 22.2 (Feb. 2011), pp. 199–210.

[Pas+19] A. Paszke et al. “PyTorch: An imperative style, high-performance deep learning library”. In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

[PBF10] T. Pevný, P. Bas, and J. Fridrich. “Steganalysis by subtractive pixel adjacency matrix”. In: *IEEE Transactions on Information Forensics and Security* 5.2 (June 2010), pp. 215–224.

[Ped+11] F. Pedregosa et al. “Scikit-learn: Machine learning in Python”. In: *Journal of Machine Learning Research* 12 (Nov. 2011), pp. 2825–2830.

[Piv13] A. Piva. “An overview on image forensics”. In: *ISRN Signal Processing* 2013 (Jan. 2013).

[PL11] X. Pan and S. Lyu. “Region duplication detection using image feature matching”. In: *IEEE Transactions on Information Forensics and Security* 5.4 (2011), pp. 857–867.

[PY10] S. J. Pan and Q. Yang. “A survey on transfer learning”. In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2010), pp. 1345–1359.

[Qiu+14] X. Qiu et al. “A universal image forensic strategy based on steganalytic model”. In: *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security*. 2014, pp. 165–170.

[RD09] G. Van Rossum and F. L. Drake. *Python 3 reference manual*. Scotts Valley, CA: CreateSpace, 2009.

[Red+20] I. Redko et al. *CO-Optimal transport*. 2020. arXiv: 2002.03731 [stat.ML].

[RJC19] P. L. C. Rodrigues, C. Jutten, and M. Congedo. “Riemannian procrustes analysis: Transfer learning for brain-computer interfaces”. In: *IEEE Transactions on Biomedical Engineering* 66.8 (Aug. 2019), pp. 2390–2401.

[RLL10] S. Ryu, M. Lee, and H. Lee. “Detection of copy-rotate-move forgery using Zernike moments”. In: *Proceedings of the International Workshop on Information Hiding*. 2010, pp. 51–65.

[Rö+19] A. Rössler et al. “FaceForensics++: Learning to detect manipulated facial images”. In: *IEEE International Conference on Computer Vision*. Jan. 2019.

[Sal+17] T. Salimans et al. *PixelCNN++: Improving the PixelCNN with discretized logistic mixture likelihood and other modifications*. 2017. arXiv: 1701.05517.

[SB11] B. Shivakumar and S. Baboo. “Detection of region duplication forgery in digital images using SURF”. In: *International Journal of Computer Science Issues* 8.4 (2011), pp. 199–205.

[SB91] M. Swain and D. Ballard. “Color indexing”. In: *International Journal of Computer Vision* 7 (Nov. 1991), pp. 11–32.

[Tra+13] D. Tralic et al. “CoMoFoD – New database for copy-move forgery detection”. In: *Proceedings of the International Symposium on Electronics in Marine*. 2013, pp. 1–6.

[UVL18] D. Ulyanov, A. Vedaldi, and V. Lempitsky. “Deep image prior”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[Vir+20] P. Virtanen et al. “SciPy 1.0: Fundamental algorithms for scientific computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272.

[WAAN17] Y. Wu, W. Abd-Almageed, and P. Natarajan. “Deep matching and validation network: An end-to-end solution to constrained image splicing localization and detection”. In: *Proceedings of the ACM International Conference on Multimedia*. 2017, pp. 1480–1502.

[WAAN18] Y. Wu, W. Abd-Almageed, and P. Natarajan. “BusterNet: Detecting copy-move image forgery with source/target localization”. In: *The European Conference on Computer Vision (ECCV)*. Sept. 2018.

[Wan+19] S. Wang et al. “CNN-generated images are surprisingly easy to spot... for now”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Dec. 2019.

[WAN19] Y. Wu, W. AbdAlmageed, and P. Natarajan. “ManTra-Net: Manipulation tracing network for detection and localization of image forgeries with anomalous features”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 9535–9544.

[WCV11] S. Van Der Walt, S. C. Colbert, and G. Varoquaux. “The NumPy array: A structure for efficient numerical computation”. In: *Computing in Science & Engineering* 13.2 (2011), p. 22.

[Yan+19] H. Yang et al. “Copy-move forgery detection based on adaptive keypoints extraction and matching”. In: *Multimedia Tools and Applications* 78 (Sept. 2019).

[Yos+14] J. Yosinski et al. “How transferable are features in deep neural networks?” In: *Proceedings of the International Conference on Neural Information Processing Systems*. 2014, pp. 3320–3328.

[Yua11] H. Yuan. “Blind forensics of median filtering in digital images”. In: *IEEE Transactions on Information Forensics and Security* 6.4 (Dec. 2011), pp. 1335–1345.

[Zan+18] P. Zanini et al. “Transfer learning: A Riemannian geometry framework with applications to brain computer interfaces”. In: *IEEE Transactions on Biomedical Engineering* 65.5 (May 2018), pp. 1107–1116.

[Zha+18] H. Zhang et al. “mixup: Beyond empirical risk minimization”. In: *International Conference on Learning Representations*. 2018.

[ZP19] J. Zhong and C. Pun. “An end-to-end Dense-InceptionNet for image copy-move forgery detection”. In: *IEEE Transactions on Information Forensics and Security* 15 (2019), pp. 2134–2146.

[ZPK15] M. Zampoglou, S. Papadopoulos, and Y. Kompatsiaris. “Detecting image splicing in the wild (WEB)”. In: *Proceedings of the IEEE International Conference on Multimedia and Expo Workshops*. 2015, pp. 1–6.

[ZW11] D. Zoran and Y. Weiss. “From learning models of natural image patches to whole image restoration”. In: *Proceedings of IEEE International Conference on Computer Vision*. 2011, pp. 479–486.