

# MEDIA FORENSICS USING MACHINE LEARNING APPROACHES

A Dissertation

Submitted to the Faculty

of

Purdue University

by

David Güera

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2019

Purdue University

West Lafayette, Indiana

**THE PURDUE UNIVERSITY GRADUATE SCHOOL**  
**STATEMENT OF DISSERTATION APPROVAL**

Dr. Edward J. Delp, Chair

School of Electrical and Computer Engineering

Dr. Mary L. Comer

School of Electrical and Computer Engineering

Dr. Amy R. Reibman

School of Electrical and Computer Engineering

Dr. Fengqing M. Zhu

School of Electrical and Computer Engineering

**Approved by:**

Dr. Dimitrios Peroulis

Head of the School Graduate Program

*“We began as wanderers, and we are wanderers still. We have lingered long enough on the shores of the cosmic ocean. We are ready at last to set sail for the stars.”*

— Carl Sagan, *Cosmos*

## ACKNOWLEDGMENTS

I would like to begin by thanking my doctoral advisor Professor Edward J. Delp. This dissertation would not even exist if he had not given me the opportunity to be a part of his laboratory while I still was an undergrad. If I have been able to achieve so much, it is because of his encouragement, support, and guidance, which I will always remember. I also want to personally thank all the members of my Ph.D. advisory committee: Professor Mary Comer for her encouragement throughout my graduate studies and her educational approach has always been welcomed. Professor Amy R. Reibman, for her guidance and support, attention to details, thoughtfulness and analytical skills that have always made working with her an enjoyable and continuous learning experience. And to Professor Fengqing Maggie Zhu for her insightful feedback and suggestions. Our conversations have always led to fruitful research ideas.

The Video and Image Processing Laboratory (VIPER) is what it is because of its people, and I would not be who I am right now if I had not been able to learn and enjoy so much from them. I would like to thank all of its formers members who all helped me in one way or another during their time at the VIPER Lab: Dr. Neeraj J. Gadgil, Dr. Khalid Tahboub, Dr. Joonsoo Kim, Dr. Yu Wang, Dr. Chichen Fu, Dr. Shaobo Fang, Dr. Javier Ribera Prat, Dr. David Joon Ho, Dr. Soonam Lee, Dr. Jeehyun Choe, Dr. Dahjung Chung, Blanca Delgado, He Li, and Chang Liu.

I would also like to thank all its current members, without whom this would have been a much more boring journey: Sriram Baireddy, Emily R. Bartusiak, Enyu Cai, Alain Chen, Di Chen, Qingshuang Chen, Yuhao Chen, Jiaqi Guo, Mridul Gupta, Shuo Han, Hanxiang (Hans) Hao, Jiangpeng He, János Horváth, Han Hu, Runyu Mao, Daniel Mas, Ruiting Shao, Zeman Shao, Changye Yang, Sri Kalyan Yarlagadda, and Yifan Zhao.



I would also like to express my gratitude to my international collaborators and friends at the Politecnico di Milano: Professor Paolo Bestagini, from whom I have learned how to become a better researcher, Professor Stefano Tubaro, Dr. Luca Bondi, Nicolò Bonettini, Sara Mandelli, Dr. Silvia Lameri, and Dr. Luca Baroffio. My gratitude also goes to all the professors and students at the University of Siena, New York University, Notre Dame University, the University of Southern California, and the University of Campinas who have been involved in the MediFor project throughout these years. It has been an honor to learn from all of you.

And to my parents and brother, Montse, Antonio, and Albert, and all my family and friends, infinite thanks to you for all the love and unconditional support. I would not be who I am without you.

This material is based on research sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under agreement number FA8750-16-2-0173. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, AFRL or the U.S. Government.

## TABLE OF CONTENTS

|   | Page |
|---|------|
| LIST OF TABLES . . . . .  | ix   |
| LIST OF FIGURES . . . . .   | x    |
| NOMENCLATURE . . . . .  | xii  |
| ABSTRACT . . . . .  | xiv  |
| 1 INTRODUCTION . . . . .  | 1    |
| 1.1 Camera Model Identification . . . . .                           | 4    |
| 1.2 Deepfake Video Detection . . . . .                              | 5    |
| 1.3 Video Manipulation Detection Using Stream Descriptors . . . . . | 7    |
| 1.4 Contributions Of This Thesis . . . . .                          | 8    |
| 1.5 Publications Resulting From This Work . . . . .                 | 8    |
| 2 CAMERA MODEL IDENTIFICATION . . . . .                             | 12   |
| 2.1 Overview . . . . .  | 12   |
| 2.2 Background On CNNs . . . . .                                    | 14   |
| 2.3 Camera Model Identification Using CNNs <sup>1</sup> . . . . .   | 17   |
| 2.3.1 Patch Selection . . . . .                                     | 17   |
| 2.3.2 CNN Training . . . . .  | 19   |
| 2.3.3 SVM Training . . . . .  | 19   |
| 2.3.4 Majority Voting . . . . .                                     | 20   |
| 2.4 Evaluation Setup . . . . .                                      | 20   |
| 2.4.1 CNN Architectures . . . . .                                   | 20   |
| 2.4.2 Training Strategies . . . . .                                 | 23   |
| 2.5 Experimental Results . . . . .                                  | 26   |
| 2.5.1 Impact Of The CNN Architecture . . . . .                      | 27   |

---

<sup>1</sup>This is joint work with Dr. Luca Bondi, Dr. Luca Baroffio, Prof. Paolo Bestagini, and Prof. Stefano Tubaro

|   | Page |
|---|------|
| 2.5.2 Impact Of Training Strategy . . . . .                           | 28   |
| 2.5.3 Comparison With The State-Of-The-Art . . . . .                  | 31   |
| 3 RELIABILITY MAP ESTIMATION FOR CAMERA IDENTIFICATION . . . . .      | 34   |
| 3.1 Overview . . . . .  | 34   |
| 3.2 Problem Statement And Related Work . . . . .                      | 36   |
| 3.2.1 Problem Formulation . . . . .                                   | 36   |
| 3.2.2 Convolutional Neural Networks In Multimedia Forensics . . . . . | 37   |
| 3.3 Patch Reliability Estimation Method <sup>2</sup> . . . . .        | 39   |
| 3.3.1 Patch Extraction . . . . .                                      | 39   |
| 3.3.2 Patch Camera Reliability . . . . .                              | 40   |
| 3.3.3 Patch Camera Attribution . . . . .                              | 41   |
| 3.3.4 Reliability-Map And Camera Attribution . . . . .                | 41   |
| 3.4 Experimental Results . . . . .                                    | 42   |
| 3.4.1 Dataset . . . . .   | 42   |
| 3.4.2 Training Strategies . . . . .                                   | 44   |
| 3.5 Discussion . . . . .  | 45   |
| 3.5.1 Patch Reliability . . . . .                                     | 45   |
| 3.5.2 Camera Model Attribution . . . . .                              | 49   |
| 4 COUNTER-FORENSICS FOR CAMERA MODEL IDENTIFICATION . . . . .         | 52   |
| 4.1 Overview . . . . .  | 52   |
| 4.2 CNN-Based Camera Model Identification . . . . .                   | 55   |
| 4.3 Proposed Method <sup>3</sup> . . . . .                            | 57   |
| 4.3.1 Fast Gradient Sign Method . . . . .                             | 58   |
| 4.3.2 Jacobian-Based Saliency Map Attack . . . . .                    | 58   |
| 4.3.3 Implementation Details . . . . .                                | 59   |
| 4.4 Experimental Results . . . . .                                    | 59   |

<sup>2</sup>This is joint work with Mr. Sri Kalyan Yarlagadda, Prof. Fengqing Maggie Zhu, Prof. Paolo Bestagini, and Prof. Stefano Tubaro

<sup>3</sup>This is joint work with Dr. Yu Wang, Dr. Luca Bondi, Prof. Paolo Bestagini, and Prof. Stefano Tubaro

|   | Page |
|---|------|
| 4.4.1 Experimental Setup . . . . .                                | 60   |
| 4.4.2 CNN Architecture . . . . .                                  | 61   |
| 4.4.3 Adversarial Image Generation . . . . .                      | 64   |
| 5 DEEPAKE VIDEO DETECTION . . . . .                               | 68   |
| 5.1 Overview . . . . .  | 68   |
| 5.2 Related Work . . . . .  | 70   |
| 5.3 Deepfake Videos Exposed . . . . .                             | 71   |
| 5.3.1 Creating Deepfake Videos . . . . .                          | 71   |
| 5.4 Recurrent Network For Deepfake Detection . . . . .            | 74   |
| 5.4.1 Convolutional LSTM . . . . .                                | 75   |
| 5.5 Experiments . . . . .   | 76   |
| 5.5.1 Dataset . . . . .   | 76   |
| 5.5.2 Parameter Settings . . . . .                                | 76   |
| 5.5.3 Results . . . . .   | 77   |
| 6 VIDEO MANIPULATION DETECTION USING STREAM DESCRIPTORS . . . . . | 80   |
| 6.1 Overview . . . . .  | 80   |
| 6.2 Related Work . . . . .  | 82   |
| 6.3 Proposed Method <sup>4</sup> . . . . .                        | 82   |
| 6.4 Experimental Results . . . . .                                | 85   |
| 6.4.1 Datasets . . . . .  | 85   |
| 6.4.2 Experimental Setup . . . . .                                | 85   |
| 6.4.3 Results and Discussion . . . . .                            | 86   |
| 7 SUMMARY AND FUTURE WORK . . . . .                               | 91   |
| 7.1 Overview . . . . .  | 91   |
| 7.2 Complete List Of Publications . . . . .                       | 93   |
| REFERENCES . . . . .  | 97   |
| VITA . . . . .  | 113  |

---

<sup>4</sup>This is joint work with Mr. Sriram Baireddy, Prof. Paolo Bestagini, and Prof. Stefano Tubaro

## LIST OF TABLES

| Table   | Page |
|---|------|
| 2.1 Summary of the CNN hyper-parameters . . . . .   | 23   |
| 2.2 Structure of the reference CNN architecture $\mathcal{M}_{\text{Conv4}}$ . . . . .        | 23   |
| 2.3 The 4 proposed CNN architectures . . . . .  | 24   |
| 2.4 Small scale example for three different splitting strategies . . . . .                    | 27   |
| 2.5 CNN classification accuracy for the different sets of <i>Fair-60</i> . . . . .            | 28   |
| 3.1 Camera model attribution accuracy using selected reliable patches only . . . . .          | 49   |
| 4.1 Number of image patches per camera class for each of the dataset splits . . . . .         | 61   |
| 4.2 Single patch accuracy results for each split of the patch dataset . . . . .               | 63   |
| 4.3 Error rate and confidence score values of trained model under untargeted attack . . . . . | 65   |
| 4.4 Error rate and confidence score values of trained model under targeted attack . . . . .   | 66   |
| 5.1 Deepfakes classification results . . . . .  | 78   |

## LIST OF FIGURES

| Figure  | Page |
|---|------|
| 1.1 Venn diagram of AI technologies . . . . .                                   | 2    |
| 1.2 Taxonomy of deep CNN architectures . . . . .                                | 3    |
| 1.3 Venn diagram of computer graphics . . . . .                                 | 5    |
| 2.1 CNN architecture . . . . .  | 16   |
| 2.2 Proposed pipeline for camera model identification . . . . .                 | 17   |
| 2.3 Patch examples and relative quality measure . . . . .                       | 18   |
| 2.4 Fair splitting policy results . . . . .                                     | 29   |
| 2.5 Fair balanced splitting policy results . . . . .                            | 30   |
| 2.6 Unfair splitting policy results . . . . .                                   | 31   |
| 2.7 Comparison between the overall pipeline and the state-of-the-art . . . . .  | 33   |
| 3.1 Reliability map representation . . . . .                                    | 35   |
| 3.2 Block diagram of the proposed approach . . . . .                            | 38   |
| 3.3 Representation of the CNN architecture working on image patches . . . . .   | 39   |
| 3.4 Examples of images and their estimated reliability maps . . . . .           | 42   |
| 3.5 Loss and accuracy curves on training and validation datasets . . . . .      | 47   |
| 3.6 Patch reliability estimation accuracy . . . . .                             | 47   |
| 3.7 ROC curves on reliable patch detection . . . . .                            | 48   |
| 3.8 Camera model attribution confusion matrix without patch selection . . . . . | 50   |
| 3.9 Camera model attribution confusion matrix with patch selection . . . . .    | 51   |
| 4.1 Example of a pipeline for camera model identification . . . . .             | 56   |
| 4.2 Block diagram of our proposed method . . . . .                              | 57   |
| 4.3 Example of images from the training set of the patch dataset . . . . .      | 62   |
| 4.4 Example of untargeted adversarial image generation attack . . . . .         | 64   |
| 4.5 Example of targeted adversarial image generation attack . . . . .           | 67   |

| Figure  | Page |
|---|------|
| 5.1 Examples of face swapping . . . . .   | 68   |
| 5.2 Deepfake generation process . . . . .   | 72   |
| 5.3 Overview of our deepfake detection system . . . . .   | 75   |
| 6.1 Examples of video stream descriptors . . . . .  | 81   |
| 6.2 Block diagram of the training stage for the video stream descriptor detector . . .  | 83   |
| 6.3 Block diagram of the testing stage for the video stream descriptor detector . . .   | 83   |
| 6.4 PR curves, F1 score, AUC score, and AP score on the test set for all the trained models using 10% of the available training data (68 videos) . . . . .  | 87   |
| 6.5 PR curves, F1 score, AUC score, and AP score on the test set for all the trained models using 25% of the available training data (169 videos) . . . . . | 88   |
| 6.6 PR curves, F1 score, AUC score, and AP score on the test set for all the trained models using 50% of the available training data (339 videos) . . . . . | 89   |
| 6.7 PR curves, F1 score, AUC score, and AP score on the test set for all the trained models using 75% of the available training data (508 videos) . . . . . | 90   |

## NOMENCLATURE

|       |   |
|-------|---|
| AI    | Artificial Intelligence                   |
| AFRL  | Air Force Research Laboratory             |
| AUC   | Area Under the Curve                      |
| CFA   | Color Filter Array                        |
| CG    | Computer Graphics                         |
| CRF   | Constant Rate Factor                      |
| CNN   | Convolutional Neural Network              |
| DARPA | Defense Advanced Research Projects Agency |
| Exif  | Exchangeable Image File Format            |
| FGSM  | Fast Gradient Sign Method                 |
| FPS   | Frames Per Second                         |
| GAN   | Generative Adversarial Network            |
| GIMP  | GNU Image Manipulation Program            |
| GPU   | Graphics Processing Unit                  |
| JPEG  | Joint Photographic Experts Group          |
| JSMA  | Jacobian-based Saliency Map Attack        |
| LSTM  | Long Short Term Memory                    |
| NGA   | National Geospatial-Intelligence Agency   |
| PR    | Precision-Recall                          |
| PRNU  | Photo Response Non Uniformity             |
| PSNR  | Peak Signal to Noise Ratio                |
| ReLU  | Rectified Linear Unit                     |
| RGB   | Red, Green, and Blue                      |
| RNN   | Recurrent Neural Network                  |



|     |                                   |
|-----|-----------------------------------|
| ROC | Receiver Operating Characteristic |
| SGD | Stochastic Gradient Descent       |
| SVM | Support Vector Machine            |

## ABSTRACT

Güera, David Ph.D., Purdue University, December 2019. Media Forensics Using Machine Learning Approaches. Major Professor: Edward J. Delp.

Consumer-grade imaging sensors have become ubiquitous in the past decade. Images and videos, collected from such sensors are used by many entities for public and private communications, including publicity, advocacy, disinformation, and deception. In this thesis, we present tools to be able to extract knowledge from and understand this imagery and its provenance. Many images and videos are modified and/or manipulated prior to their public release. We also propose a set of forensics and counter-forensic techniques to determine the integrity of this multimedia content and modify it in specific ways to deceive adversaries. The presented tools are evaluated using publicly available datasets and independently organized challenges.

## 1. INTRODUCTION

The popularization of the smartphone has radically changed how we interact with each other. These devices have enabled unprecedented levels of generation of digital content such as images and videos. This digital content is now seen as the gold standard to store our memories or signal our attendance to social events. Due to the importance placed on images and videos and their major role in social networks and journalistic media, numerous pieces of software have been developed to edit and refine these digital assets. However, this software innovation has come at a price. Doctoring images and videos has become common practice, either done for aesthetic or malicious purposes. The prevalence of these forged digital media is leading the public to stop blindly accepting any kind of image or video as digital evidence. Zhu and Farid [1, 2] were among the first to hint at this trend. This mistrust is also being exacerbated by the “fake news” [3] phenomenon and by the recent popularization of methods based on machine learning techniques such as generative adversarial networks [4] that can be exploited by malicious actors.

To mitigate the spread of misinformation and aid law enforcement agencies carry out investigations in which digital assets may be involved, the research community has been developing tools to verify and authenticate such content. The field of digital image and video forensics was born out of these efforts [5]. In its current form, digital image and video forensics include two large research areas which are the focus of this work: source image forensics and content-based forensics. To tackle the problem of source image forensics, we particularize it as source camera identification task, which we propose to solve using CNN-based methods [6–8]. Content-based forensics includes computer graphics forensic, for which we focus on the recent Deepfakes video manipulation technique [9, 10]. We also study adversarial content image generation and detection for CNN camera model identification methods [11]. Finally, we also investigate the use of video metadata analysis for manipulation detection [12].

Initially, the problem of source image forensics was addressed through methods that rely on the study of the statistics of the image and other pattern recognition techniques. Due to the improvements in classification methods shown by Krizhevsky *et al.* [13] in 2012, coupled with the increasing availability of large computational resources, researchers have shown a major interest in techniques based on CNNs. Deep learning based methods have been successfully used in the field of source image forensics, and have proved their effectiveness in various competitions [14, 15]. To better understand the results achieved with deep learning methods, we give a comprehensive review of those approaches in the following lines, and how they relate to each other, as shown in Figure 1.1 This will allow any reader of this work to better understand the context in which we have developed our solutions.

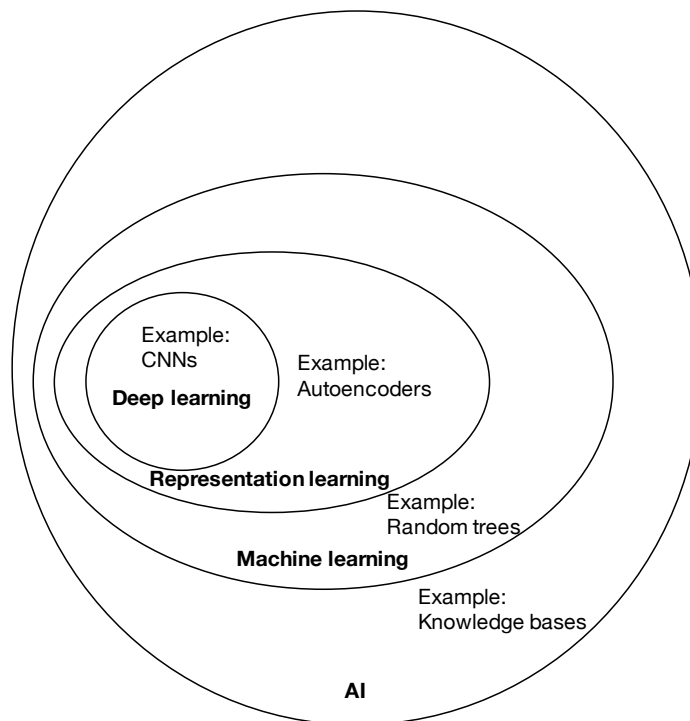


Fig. 1.1.: A Venn diagram showing how deep learning is a subset of representation learning, which in turn is a subset of machine learning, which is used for many but not all approaches to AI. Each section of the Venn diagram includes an example of an AI technology used in this work. Adapted from Goodfellow *et al.* [16].

A series of breakthroughs in 2006 [17–19] made the use of deep neural networks viable and gave rise to the field known as deep learning [16, 20]. Following methods based on deep learning have consistently achieved remarkable results on a series of tasks such as handwritten text recognition, video scene understanding, and image classification. To solve these tasks, specialized techniques have been developed in the field of computer vision, including convolutional neural networks, recurrent neural networks, and generative adversarial networks. Among them, CNNs and RNNs have been shown to be effective when dealing with image and video related tasks, and have been subsequently adopted as a basis for numerous digital forensics methods.

In broad terms, the basic components of CNNs consist mainly of convolutional layers, pooling layers, and activation functions. To construct the architecture of a convolutional neural network, these layers are generally stacked together. The main recent advances on the building and training of CNNs can be grouped in: structural reformulation, parameters optimizations, regularization, and loss function. Among those, structural reformulation plays the most important role in improving the performance. Typical CNN architectures such as AlexNet [13], VGG [21], GoogleNet [22], ResNet [23], DenseNet [24], Xception [25], SENet [26], Siamese Network [27], are well known. Figure 1.2 shows the overall taxonomy of the aforementioned architectures. We refer the interested reader to the review by Goodfellow *et al.* [16], Khan *et al.* [28], and Gu *et al.* [29] for further details about deep learning.

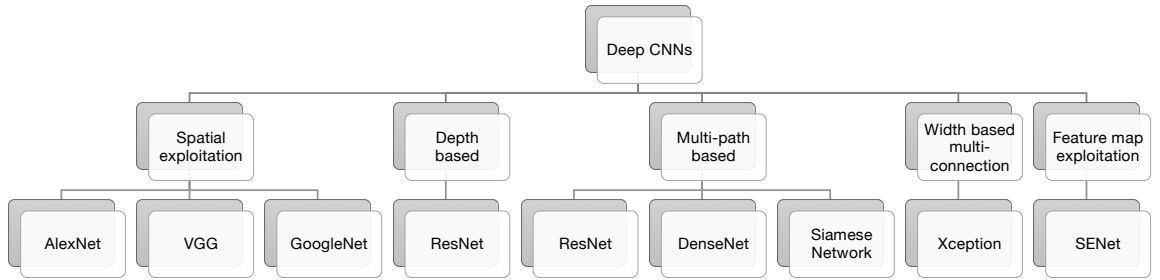


Fig. 1.2.: Taxonomy of deep CNN architectures. Adapted from Khan *et al.* [28].

## 1.1 Camera Model Identification

Camera model identification is paramount to verify image origin and authenticity in a blind fashion. State-of-the-art techniques leverage the analysis of features describing characteristic footprints left on images by different camera models from the image acquisition pipeline (e.g., traces left by proprietary demosaicing strategies, etc.).

Motivated by the very accurate performance achieved by feature-based methods, as well as by the progress brought by deep architectures in machine learning, we explore in Chapter 2 the possibility of taking advantage of convolutional neural networks (CNNs) for camera model identification. More specifically, we investigate: (i) the capability of different network architectures to learn discriminant features directly from the observed images; (ii) the dependency between the amount of training data and the achieved accuracy; (iii) the importance of selecting a correct protocol for training, validation and testing. Our study shows that promising results can be obtained on small image patches training a CNN with an affordable setup (i.e., a personal computer with one dedicated GPU) in a reasonable amount of time (i.e., approximately one hour), given that a sufficient amount of training images is available.

However, some patches of an image may not contain enough information related to the camera model (e.g., saturated patches). In Chapter 3, we propose a CNN-based solution to estimate the camera model attribution reliability of a given image patch. We show that we can estimate a reliability-map indicating which portions of the image contain reliable camera traces. Testing using a well known dataset confirms that by using this information, it is possible to increase small patch camera model attribution accuracy by more than 8% on a single patch.

In Chapter 4 we revisit the problem of identifying the camera model or type that was used to take an image and how it can be spoofed. Due to the linear nature of CNNs and the high-dimensionality of images, neural networks are vulnerable to attacks with adversarial examples. These examples are imperceptibly different from correctly classified images but are misclassified with high confidence by CNNs. We describe a counter-forensic method

capable of subtly altering images to change their estimated camera model when they are analyzed by any CNN-based camera model detector. Our method can use both the Fast Gradient Sign Method (FGSM) or the Jacobian-based Saliency Map Attack (JSMA) to craft these adversarial images and does not require direct access to the CNN. Our results show that even advanced deep learning architectures trained to analyze images and obtain camera model information are still vulnerable to our proposed method.

## 1.2 Deepfake Video Detection

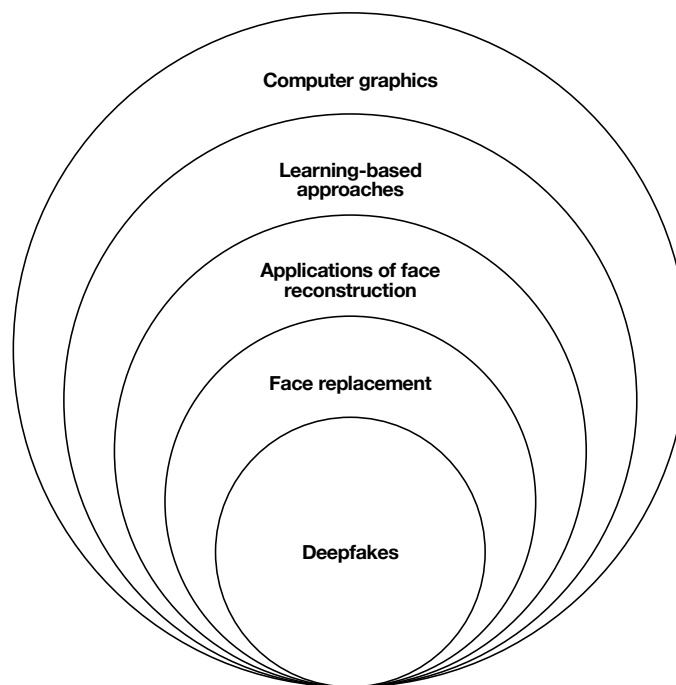


Fig. 1.3.: A Venn diagram showing how Deepfakes is an instance of face replacement, which in turn is a particular application of face reconstruction, which belong to the family of learning-based computer graphics methods.

Re-constructing, tracking, and analyzing human faces based on visual input remains an open problem. Several methods to tackle these specific problems have been proposed by the field computer vision and graphics. While convincing manipulations of digital images and videos have been demonstrated for several decades through the use of visual effects, recent advances in deep learning have led to a dramatic increase in the realism of fake

content and the accessibility in which it can be created [30–35]. More specifically, several approaches for face replacement or swapping have been proposed in the literature, either model-based [36, 37], pure image-based [38], or learning-based [39]. These techniques swap the face of a target person in a video with the face of a source person, where the videos may be recorded under different illumination conditions. Synthesizing a believable video sequence that is realistic and consistent in the temporal sense remains unsolved. In recent months a learning-based free computer graphics tool has made it easy to create believable face swaps in videos that leaves few traces of manipulation, in what are known as “deepfake” videos. We refer the readers to Figure 1.3 and the excellent overview of face reconstruction, tracking, and applications by Zollhöfer *et al.* [40] to better understand how deepfakes fit within the set of computer graphics techniques.

Scenarios where these realistic fake videos are used to create political distress, blackmail someone or fake terrorism events are easily envisioned. Hence, the main advances of the multimedia forensic community regarding videos have focused on detecting manipulations that are computationally cheap to generate, such as dropped or duplicated frames [41], chroma-key compositions [42], varying interpolation types [43], or copy-move manipulations [44]. Further studies explicitly focus on detecting facial manipulations, such as distinguishing computer generated faces from natural ones [45], morphed faces [46], face splicing [47], face swapping [48], and DeepFakes [10]. For face manipulation detection, some approaches exploit artifacts that originate during the synthesis phase, such as eye blinking [49], or color, texture and shape cues [47]. More general solutions propose a deep network trained to capture the subtle inconsistencies arising from low and high level features [50]. For a more in-depth analysis of the space of face-based manipulations and detectors, we suggest the readers the excellent review by Rössler *et al.* [51].

In Chapter 5 we propose a temporal-aware pipeline to automatically detect deepfake videos. Our system uses a convolutional neural network to extract frame-level features. These features are then used to train a recurrent neural network (RNN) that learns to classify if a video has been subject to manipulation or not. We evaluate our method against a large



set of deepfake videos collected from multiple video websites. We show how our system can achieve competitive results in this task while using a simple architecture.

### 1.3 Video Manipulation Detection Using Stream Descriptors

Recent digital forensics literature has revealed that a video file structure contains a lot of information about the history of its content. At the same time, this structure is much more difficult to extract and modify for a user than metadata, since available editing software and metadata editors do not have a functionality to modify such a low-level information, like the internal order of the core file structures. In [52], Gloe *et al.* explore the low-level characteristics represented by metadata and low-level file format information, with an emphasis on the structure of the video file container. Indeed, video standards prescribe only a limited number of characteristics for the data container formats, thus leaving a lot of discretion to the manufacturer. This lead to differences that can be exploited for forensic purposes. However, while providing a pioneering exploration of video container formats from a forensic viewpoint, the manual approach proposed in [52] reduces the forensic capabilities when the containers may be huge, deeply nested and strongly variable among different models.

Finally, in Chapter 6, we show how simple machine learning classifiers can be highly effective at detecting video manipulations when the appropriate data is used, completely avoiding the pixel space. Up until now, most video manipulation detection techniques have focused on analyzing the pixel data to spot forged content. Furthermore the typicality of some container features analysis can be hardly quantified by manual inspection. Using well-known datasets, our results show that this scalable approach can achieve a high manipulation detection score if the manipulators have not done a careful data sanitization of the multimedia stream descriptors. More specifically, we use an ensemble of a random forest and an SVM trained on multimedia stream descriptors from both forged and pristine videos. With this approach, we have achieved an extremely high video manipulation detection score while requiring very limited amounts of data.

## 1.4 Contributions Of This Thesis

- We explore the possibility of leveraging convolutional neural networks for camera model identification.
- We show that we can estimate a reliability-map indicating which portions of an image contain reliable camera traces for its identification.
- We describe a counter-forensic method capable of subtly altering images to change their estimated camera model.
- We propose a temporal-aware pipeline to automatically detect Deepfake videos.
- We demonstrate how by using the structure present in the stream descriptors of videos we can quickly determine if they have been manipulated without having to examine their pixel content.

## 1.5 Publications Resulting From This Work

1. L. Bondi, L. Baroffio, **D. Güera**, P. Bestagini, E. J. Delp, S. Tubaro, “First Steps Toward Camera Model Identification With Convolutional Neural Networks”, *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 259-263, December 2016.  
URL: <https://doi.org/10.1109/LSP.2016.2641006>
2. L. Bondi, **D. Güera**, L. Baroffio, P. Bestagini, E. J. Delp, S. Tubaro, “A Preliminary Study on Convolutional Neural Networks for Camera Model Identification”, *Proceedings of the IS&T Electronic Imaging*, vol. 2017, no. 7, pp. 67-76, Burlingame, CA, January 2017.  
URL: <https://doi.org/10.2352/ISSN.2470-1173.2017.7.MWWSF-327>
3. L. Bondi, S. Lameri, **D. Güera**, P. Bestagini, E. J. Delp, S. Tubaro, “Tampering Detection And Localization Through Clustering Of Camera-Based CNN Features”,

*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1855-1864, Honolulu, HI, July 2017.

URL: <https://doi.org/10.1109/CVPRW.2017.232>

4. **D. Güera**, Y. Wang, L. Bondi, P. Bestagini, S. Tubaro, E. J. Delp, “A Counter-Forensic Method For CNN-Based Camera Model Identification”, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1840-1847, Honolulu, HI, July 2017.

URL: <https://doi.org/10.1109/CVPRW.2017.230>

5. S. K. Yarlagadda, **D. Güera**, P. Bestagini, F. Zhu, S. Tubaro, E. J. Delp, “Satellite Image Forgery Detection and Localization Using GAN and One-Class Classifier”, *Proceedings of the IS&T Electronic Imaging*, vol. 2018, no. 7, pp. 214-1-214-9, Burlingame, CA, January 2018.

URL: <https://doi.org/10.2352/ISSN.2470-1173.2018.07.MWSF-214>

6. **D. Güera**, S. K. Yarlagadda, P. Bestagini, F. Zhu, S. Tubaro, E. J. Delp, “Reliability Map Estimation For CNN-Based Camera Model Attribution”, *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, pp. 964-973, Lake Tahoe, NV, February 2018.

URL: <https://doi.org/10.1109/WACV.2018.00111>

7. N. Bonettini, L. Bondi, **D. Güera**, S. Mandelli, P. Bestagini, S. Tubaro, E. J. Delp, “Fooling PRNU-Based Detectors Through Convolutional Neural Networks”, *Proceedings of the IEEE European Signal Processing Conference*, Rome, Italy, September 2018.

URL: <https://doi.org/10.23919/EUSIPCO.2018.8553596>

8. **D. Güera**, E. J. Delp, “Deepfake Video Detection Using Recurrent Neural Networks”, *Proceedings of the IEEE International Conference on Advanced Video and*

*Signal-based Surveillance*, Auckland, New Zealand, November 2018.

URL: <https://doi.org/10.23919/EUSIPCO.2018.8553596>

9. E. R. Bartusiak, S. K. Yarlagadda, **D. Güera**, F. Zhu, P. Bestagini, S. Tubaro, E. J. Delp, “Splicing Detection And Localization In Satellite Imagery Using Conditional GANs”, *Proceedings of the IEEE International Conference on Multimedia Information Processing and Retrieval*, San Jose, California, March 2019.  
URL: <https://doi.org/10.1109/MIPR.2019.00024>
10. S. K. Yarlagadda, **D. Güera**, D. Mas Montserrat, F. Zhu, P. Bestagini, S. Tubaro, E. J. Delp, “Shadow Removal Detection And Localization For Forensics Analysis”, *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Brighton, UK, May 2019.  
URL: <https://doi.org/10.1109/ICASSP.2019.8683695>
11. **D. Güera**, S. Baireddy, P. Bestagini, S. Tubaro, E. J. Delp, “We Need No Pixels: Video Manipulation Detection Using Stream Descriptors”, *Proceedings of the International Conference on Machine Learning, Synthetic Realities: Deep Learning for Detecting AudioVisual Fakes Workshop*, Long Beach, California, June 2019.  
URL: <https://arxiv.org/abs/1906.08743>
12. J. Horváth, **D. Güera**, S. K. Yarlagadda, P. Bestagini, F. Zhu, S. Tubaro, E. J. Delp, “Anomaly-based Manipulation Detection in Satellite Images”, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Workshop on Media Forensics*, Long Beach, California, June 2019.  
URL: [http://openaccess.thecvf.com/content\\_CVPRW\\_2019/html/Media\\_Forensics/Horvath\\_Anomaly-Based\\_Manipulation\\_Detection\\_in\\_Satellite\\_Images\\_CVPRW\\_2019\\_paper.html](http://openaccess.thecvf.com/content_CVPRW_2019/html/Media_Forensics/Horvath_Anomaly-Based_Manipulation_Detection_in_Satellite_Images_CVPRW_2019_paper.html)
13. N. Bonettini, **D. Güera**, L. Bondi, P. Bestagini, E. J. Delp, S. Tubaro, “Image Anonymization Detection With Deep Handcrafted Features”, *IEEE International*

Conference on Image Processing (ICIP), Taipei, Taiwan, September 2019.

URL: <https://doi.org/10.1109/ICIP.2019.8804294>

## 2. CAMERA MODEL IDENTIFICATION

### 2.1 Overview

The rapid proliferation of inexpensive image capturing devices has driven the widespread diffusion of digital pictures on the web. Sharing any type of images through websites and social media is now an option everywhere. Verifying the veracity and authenticity of these widely distributed (and re-distributed) images is far from being an easy task [5, 53]. For this reason, the multimedia forensic community has investigated methodologies to assess the trustworthiness of digital images in the last few years [54, 55].

Among the many investigated forensic issues, great attention has been devoted to camera model identification [56–58]. Indeed, detecting the camera model used to take a picture can be crucial for criminal investigations and trials. This information can be exploited for solving copyright infringement cases, as well as indicating the authors of illicit material (e.g., child pornography). Even when deeper source identification granularity is needed (i.e., detecting the unique camera instance rather than just the make and model), camera model identification can be considered an important preliminary step to reduce the set of camera instances [58]. Moreover, being able to detect the camera model by analyzing small image patches is a possible way to expose splicing operations [59].

The rationale behind blind state-of-the-art camera model identification detectors is that each camera model performs peculiar operations on each image at acquisition time (e.g., different JPEG compression schemes, proprietary algorithms for CFA demosaicing, etc.). These operations leave on each picture characteristic “footprints” that can be exploited as an asset to reverse-engineer the camera model identity.

Following this idea, some methods focus on capturing characteristic footprints left during one specific step of the image acquisition pipeline. As an example, in [57] noise traces left by sensors on acquired images are exploited. Conversely, in [60] an algorithm tailored

to detect lens distortion is proposed. In [61–63], traces relative to the used demosaicing strategy are investigated. Alternatively, the effect of dust traces on image sensors is studied in [64].

Given the difficulty of properly modeling complex chains of operations typical of the image acquisition pipeline, other camera model identification methods exploit features mainly capturing statistical image properties paired with machine learning classifiers, rather than focusing on specific processing operations. As an example, a technique based on local binary patterns is proposed in [65]. More recently, the authors of [66–68] exploit pixel co-occurrence statistics computed in different domains fed to a variety of supervised classification techniques. These methods guarantee very accurate results, especially on full resolution images that provide sufficient pixel statistics.

A common aspect of all the aforementioned algorithms is that they rely on manually defined procedures to expose traces characterizing different camera models. This means that they rely on some model assumptions a priori made. However, recent advancements established by deep learning techniques in computer vision [69] have shown that it is possible to improve the accuracy in detection and classification tasks by taking advantage of great amount of data in order to learn characteristic features directly from the data itself. These methods are known as data-driven, as they learn directly from data rather than following an analytic model.

Considering that the feature learning paradigm has recently proved fruitful for forensics applications [70–73], we investigate the use of feature learning techniques in the camera model identification context, further investigating our first exploratory solutions [6, 74]. Our objective is to show that it is possible to use convolutional neural networks to learn discriminant features directly from the observed known images, rather than relying on hand-crafted descriptors. In principle, this enables to possibly capture also characteristic traces left by non-linear and difficult to model operations during the acquisition pipeline.

To conduct our study, we investigate the behavior of different CNN architectures to select a proper network for discriminant feature learning on  $64 \times 64 \times 3$  (i.e., height  $\times$  width  $\times$  colors) image patches, while keeping computational complexity at bay. In particular, we

compare a series of CNN architectures differing in the number of convolutional, pooling, inner product and rectified linear unit (ReLU) layers. For each type of architecture, several hyper-parameters choices (e.g., kernel size, stride, number of feature maps) are examined.

We focus on the importance of a proper training protocol, which is essential to make sure that the CNN learns important characteristics (e.g., properties discriminating camera models) rather than biased information (e.g., the semantic of the captured scenes). To this purpose, strongly inspired by [58], we consider different amounts of training images, depicting either the same or different scenes, and show how different training choices affect classification results.

We first report some background on CNNs. Then, we show the algorithmic pipeline devised to perform camera model identification using CNNs. Afterwards, we report all the details about the experimental setup, from the tested CNN architectures to the used dataset splits. Then, we report the numeric results achieved through our study in order to evaluate the different tested setups. Finally, we wrap up our final considerations.

## **2.2 Background On CNNs**

In this section, we provide a brief overview of convolutional neural networks sufficient to understand the rest of this thesis. For a more in depth description, please refer to one of the many available tutorial in the literature [69, 75].

Deep learning and in particular CNNs have shown very good performance in several computer vision applications such as image classification, face recognition, pedestrian detection and handwriting recognition [75]. A CNN is a complex computational model partially inspired by the human neural system that consists of a very high number of interconnected nodes, or neurons. Connections between nodes have a numeric weight parameter that can be tuned based on experience, so that the model is able to learn complex functions. The nodes of the network are organized in multiple stacked layers, each performing a simple operation on the input. The set of operations in a CNN typically comprises convolution, intensity normalization, non-linear activation and thresholding and local pooling. By min-



imizing a cost function at the output of the last layer, the weights of the network (e.g., the values of the filters in the convolutional layers) are tuned so that they are able to capture patterns in the input data and automatically extract distinctive features.

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

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

1. To define the metaparameters of the CNN, i.e., the sequence of operations to be performed, the number of layers, the number and shape of the filters in convolutional layers, etc;
2. To define a proper cost function to be minimized during the training process;
3. To prepare a (possibly large) dataset of training and test images, annotated with labels according to the specific tasks (i.e., camera models in our work).

Figure 2.1 shows a minimalistic example of a small CNN architecture depicting some of the commonly used layers. To better understand the role of each layer, we describe the most common building block:

- *Convolution*: each convolution layer is a filterbank, whose filters impulse response  $h$  are learned through training. Given an input signal  $x$ , the output of each filter is  $y = x * h$ , i.e., the valid part of the linear convolution. Convolution is typically done on 3D representations consisting of the spatial coordinates  $(x, y)$  and the number of feature maps  $p$  (e.g.,  $p = 3$  for an RGB input).
- *Max pooling*: returns the maximum value of the input  $x$  evaluated across small windows (typically of 3x3 pixels).
- *ReLU*: Rectified Linear Unit (ReLU) uses the rectification function  $y = \max(0, x)$  to the input  $x$ , thus clipping negative values to zero [76].
- *Inner Product*: indicates that the input of each neuron of the next layer is a linear combination of all the outputs of the previous layer. Combination weights are estimated during training. The dropout rate indicates the percentage of nodes that are randomly neglected during training in order to avoid data overfitting [77].
- *SoftMax*: “squashes” the input values in the range  $[0, 1]$  and guarantees that they sum up to one. This is particularly useful at the end of the network in order to interpret its outputs as probability values.

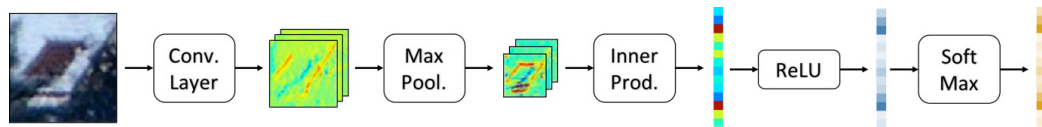


Fig. 2.1.: Simple CNN architecture consisting of commonly used layers. A small color image patch is processed through convolutional, max pooling, inner product and ReLU layers. Finally, SoftMax is used to obtain a probability vector.

## 2.3 Camera Model Identification Using CNNs<sup>1</sup>

In this chapter, we consider camera model identification as a closed set classification problem. In other words, given an image  $I$  under analysis, the goal is to detect which camera model among a set of known  $N$  ones has been used to shoot the photograph.

In order to solve this problem, our proposed method follows the pipeline depicted in Figure 2.2: (i) images are split into patches; (ii) a CNN is first trained, then used to extract meaningful features from each patch; (iii) a set of support vector machines (SVMs) are trained and used to classify each patch; (iv) a final voting procedure is used to take decision at image level aggregating patches scores.

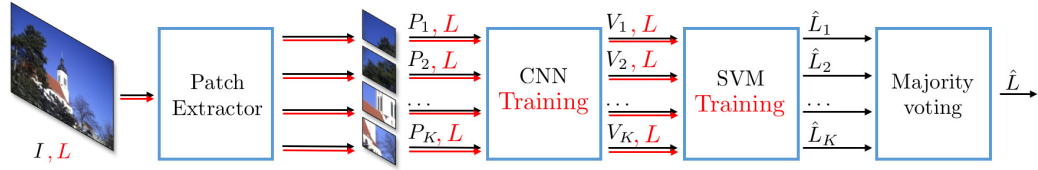


Fig. 2.2.: Proposed pipeline for camera model identification with training steps highlighted in red. During training, patches are extracted from each training image  $I$  inheriting the same label  $L$  of the image. These are used for CNN and SVM training. During evaluation, for each patch  $P_i$  of the image  $I$  under analysis, a feature vector  $V_i$  is extracted through the CNN. Feature vectors are input into a set of trained linear SVM classifiers in order to associate a candidate label  $\hat{L}_i$  to each vector. The predicted label  $\hat{L}$  for image  $I$  is obtained by majority voting.

In the following we report details about each step, leaving to the next section the description of the tested CNN architectures, which is object of investigation in this chapter.

### 2.3.1 Patch Selection

The first step of the proposed pipeline consists in splitting each image  $I$  into a set of  $K$  non-overlapping patches  $P_k$ ,  $k \in [1, K]$  of size  $64 \times 64 \times 3$  (i.e., height  $\times$  width  $\times$  color). The rationale behind this choice is twofold: (i) splitting images into patches allows us to obtain a greater amount of data for CNN training; (ii) feeding the CNN with smaller data

<sup>1</sup>This is joint work with Dr. Luca Bondi, Dr. Luca Baroffio, Prof. Paolo Bestagini, and Prof. Stefano Tubaro

(i.e., a patches rather than full resolution images) enables working with smaller and lighter CNN architectures.

However, not all patches contain enough statistical information about the used camera model. For instance, it is clear that saturated patches should not be considered during either training or testing. Therefore, we devised a patch selection procedure. Specifically, for each patch  $P_k$  within an image, we compute a quality value defined as

$$Q(P_k) = \frac{1}{3} \sum_{c \in [R, G, B]} [\alpha \cdot \beta (\mu_c - \mu_c^2) + (1 - \alpha) (1 - e^{\gamma \sigma_c})], \quad (2.1)$$

where  $\alpha$ ,  $\beta$  and  $\gamma$  are empirically set constants (set to 0.7, 4 and  $\ln(0.01)$  in our experiments), whereas  $\mu_c$  and  $\sigma_c$ ,  $c \in [R, G, B]$  are the average and standard deviation of red, green and blue components (in range  $[0, 1]$ ) of patch  $P_k$ , respectively. This quality measure tends to be lower for overly saturated or flat patches, whereas it is higher for textured patches showing some statistical variance (as shown in Figure 2.3). Therefore, we select for each image  $K$  patches with the highest  $Q$  values.

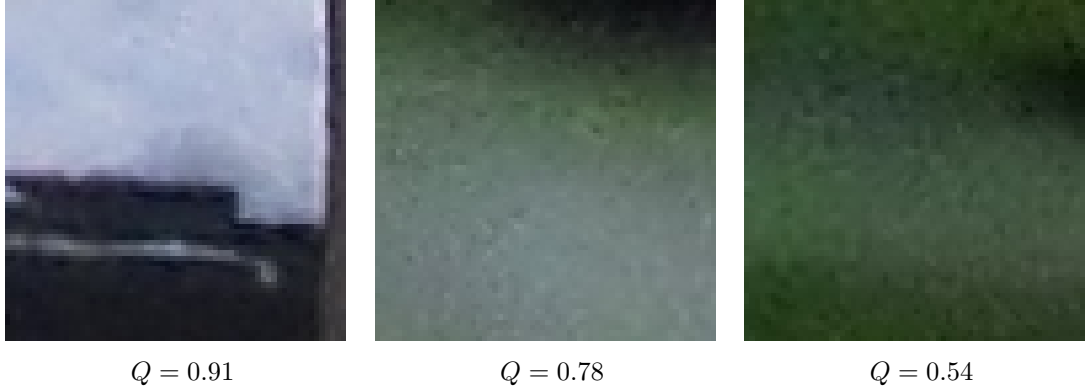


Fig. 2.3.: Patch examples and relative quality measure.

### 2.3.2 CNN Training

Given a set of training labeled images coming from  $N$  known camera models, we split them into patches and associate to each patch the same label  $L$  of the source image to which patches belong. Then, we input all available patch-label pairs for training into the CNN.

The choice of the CNN architecture is a delicate step. As an example, a too deep network may be unnecessary long to train and may contain too many parameters that need a huge training dataset to be properly tuned. However, smaller networks may not achieve accurate enough performance, losing the ability of well discriminating the used camera models. Moreover, despite the number of used layers, also the choice of filters size and stride plays a crucial role, not to mention the use of inner product layers. For these reasons in this chapter we tested different CNN architectures, the detailed description of which is left to the next section.

A common aspect among all tested architectures is that they accept as input patches of size  $64 \times 64 \times 3$ . The pixel-wise average over the training set is subtracted to each input patch. At the end of the training step, we obtain the CNN model  $\mathcal{M}$ .

### 2.3.3 SVM Training

Even though we could perform classification by simply picking the class corresponding to the maximum value from SoftMax layer, we decided to make use of an additional classification tool. Therefore, for each patch, the selected CNN model  $\mathcal{M}$  is used to extract a feature vector, stopping the forward propagation at the last-minus-one layer (before the classification inner-product layer). Feature vectors associated to training patches are used to train a battery of  $N \cdot (N - 1)/2$  linear binary SVM classifiers  $\mathcal{S}$  in a One-versus-One fashion.

### 2.3.4 Majority Voting

When a new image  $I$  is under analysis, the camera model used to acquire it is estimated as follows. A set of  $K$  patches is obtained from image  $I$  according to the quality measures defined in (2.1). Each patch  $P_k$  is processed by the trained CNN model  $\mathcal{M}$  in order to extract a feature vector  $V_k$ . The set  $\mathcal{S}$  of linear SVMs assigns a label  $\hat{L}_k$  to each patch by attributing vectors  $V_k$  to one of the available  $N$  classes (i.e., camera models). The predicted model  $\hat{L}$  for image  $I$  is obtained through majority voting on  $\hat{L}_k, k \in [1, K]$ . In case of par, random selection between equally likely models is operated.

## 2.4 Evaluation Setup

Considering the empirical nature of the proposed pipeline, it is essential to investigate through testing the impact of different CNN architectures and training strategies. In this section, we first report a detailed description of all tested CNNs. Then, we introduce the problem of biased training and describe the strategies that we tested to ensure a fair training process.

### 2.4.1 CNN Architectures

Designing a proper CNN architecture for our camera model identification pipeline is a critical step. The overall accuracy of the system is significantly determined by the extracted feature vectors from each image patch.

There are several key design choices that have to be considered as they determine the final structure of the CNN. The depth of the network, the use of pooling layers and the size of the kernels are examples and are referred to as hyper-parameters. Tuning hyper-parameters is approached in a trial-and-error fashion as there are no hard-quantitative rules that can be followed. This is due to the fact that we approach camera model identification as a data-driven problem and the final architecture of the network depends on the type of

data under consideration. In our particular case, we explore networks that accept input patches of size  $64 \times 64 \times 3$ .

In [74], we presented a CNN used to extract features for camera model identification. A summary of the hyper-parameters used in that CNN is shown in Table 2.1. The encouraging results demonstrated with that particular architecture motivated us to improve its design in [6]. In particular, we used some of the CNN architecture design guidelines proposed in [78] and proposed a new CNN architecture, hereinafter denoted as  $\mathcal{M}_{Conv4}$ . In Table 2.2 we summarize its hyper-parameters. In this chapter, we study the behavior of  $\mathcal{M}_{Conv4}$  compared to other 3 new networks, which vary in depth as will be explained in this section. In particular, we consider  $\mathcal{M}_{Conv4}$  as the base CNN on top of which we develop the 3 remaining deeper architectures.

The changes made in  $\mathcal{M}_{Conv4}$  with respect to our first proposed CNN aim to reduce the computational complexity and improve the accuracy. As suggested in [78], in order to keep the computational complexity at bay we use more convolutional layers with smaller kernel sizes instead of using larger kernels and fewer convolutional layers (e.g. 2 stacks of  $3 \times 3$  convolutional layers vs. a single  $7 \times 7$  convolutional layer). A similar idea was also proposed by Chatfield et al. in [79]. This change coupled with the reduction in the overall number of convolutional filters aim to be parameter efficient. It also has the added benefit that our convolutional neural network is able to represent more complex functions as we add more layers.

We also changed the kernel size of the pooling layers to the most conventional value of  $2 \times 2$  and changed the stride value to 2. We also added an additional pooling layer. The main idea behind having more pooling layers stems from the fact that the exact location of a feature in the original input patch (i.e. a high activation value occurs) is not as important as its relative location to other features. These layers drastically reduce the spatial dimension of the input volume they receive, which serves two purposes:

- The number of parameters is reduced by 75%, which translates into an efficient use of computer resources.

- Controlling overfitting, which happens when a model is excessively fine-tuned to the training examples and it is not able to generalize well for the validation and evaluation sets (e.g. if the number of parameters of the network is high enough, the network could just memorize the training examples).

Finally, because we still want our CNN to be able to model non-linear functions we use a single ReLU layer towards the end of the network. This will make the CNN applicable to a wide range of camera models due to the fact that the non-linearity can be helpful to capture non-trivial classes.

It was shown by the Oxford VGG team in [21] and Szegedy et al. in [22], the representational capacity of a network is largely determined by its depth. This insight was also verified with the recent development of deep residual networks [23], which can have more than 150 layers and were used by the winners of the ILSVRC-2015 competition [80]. Keeping simplicity in mind, and following the strategy that Simonyan et al. devised in [21] to prepare their submission for the ILSVRC-2014, we explored a single family of networks of increasing depth. By doing so, we took advantage of the key design choices made in our base CNN model,  $\mathcal{M}_{Conv4}$ , such as the stacks of convolutional and pooling layers structure and the kernel sizes of the convolutional layers. A stride value of 1 was chosen for the convolutional layers. This results in no skipping (i.e. our filters are applied to all the values of the input volumes they receive).

The CNN architectures, evaluated in this work, are outlined in Table 2.3, one per column.  $\mathcal{M}_{Conv4}$  was introduced earlier in [6] and we refer to the three remaining networks as  $\mathcal{M}_{Conv6}$ ,  $\mathcal{M}_{Conv8}$  and  $\mathcal{M}_{Conv10}$ . As mentioned earlier, the 4 networks vary only in the depth: from 6 weight layers in the network  $\mathcal{M}_{Conv4}$  (4 convolutional and 2 Inner Product layers) to 12 weight layers in the network  $\mathcal{M}_{Conv10}$  (10 convolutional and 2 Inner Product layers). The number of filters of each convolutional layer is rather small, starting from 32 in the first layer and then adding 16 more filters after each pooling layer, except for the last one, where we increase the number of filters by a factor of 2 reaching a total of 128 filters.



Table 2.1.: Summary of the hyper-parameters of the first CNN architecture that we presented as a feature extractor for camera model identification in [74].

| Layer          | Kernel size  | Num. filters |
|----------------|--------------|--------------|
| Conv-1         | $7 \times 7$ | 128          |
| ReLU-1         | -            | -            |
| Pool-1         | $3 \times 3$ | -            |
| Conv-2         | $7 \times 7$ | 512          |
| ReLU-2         | -            | -            |
| Pool-2         | $3 \times 3$ | -            |
| Conv-3         | $6 \times 6$ | 2048         |
| ReLU-3         | -            | -            |
| InnerProduct-1 | -            | 2048         |
| ReLU-4         | -            | -            |
| InnerProduct-2 | -            | 2048         |
| SoftMax        | -            | -            |

Table 2.2.: Structure of the reference CNN architecture  $\mathcal{M}_{\text{Conv4}}$ .  $N$  is the number of training classes. Feature are extracted after the ReLU-1 layer.

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

## 2.4.2 Training Strategies

As discussed in [58], the training procedure for machine learning-based camera model identification algorithms must be devised with great attention. While it is important to

Table 2.3.: The 4 proposed CNN architectures (shown in columns). Added layers are shown in bold and the number of filters for each convolutional layer is shown in parenthesis.

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

ensure a sufficient amount of training data, training data cannot be randomly selected. Training data must be carefully chosen in order to avoid over fitting and ensure a wide variety of images covering different scenarios.

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

In order to consider this important issue, in our study we consider the Dresden Image Dataset [81] as reference, as suggested in [58]. This dataset is composed by 73 camera devices from 25 camera models and 14 camera brands. For each device a variable number of shots has been taken in several geographical positions. For each position a set of different motives is shot. Details about the acquisition process are available at [81]. In the following we will refer to *scene* when considering the combination of a geographical position with a specific motive. This results in a total amount of 83 available scenes. We only consider camera models represented by more than one device, in order to ensure that the CNN learns model specific artifacts rather than instance specific ones. This leads to a dataset composed of 18 camera models (as Nikon D70 and D70s basically differ only in their on-device screen), for nearly 15,000 shots.

We need to split images into three different datasets: (i) a training set  $\mathbb{D}_T$  used for updating CNNs and SVMs parameters; (ii) a validation set  $\mathbb{D}_V$  used to decide the stopping point of the training step and avoid over-fitting (i.e., typically the training process is stopped when validation loss, given by SoftMax layer, reaches its minimum); (iii) an evaluation set  $\mathbb{D}_E$  used to test the trained architectures. Following the ideas presented by Kirchner et al. [58]

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

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

1. *Fair- $N_T$* : training and validation shots are split according to the depicted scene. The number of training scenes is set to  $N_T$  and shots coming from a specific scene are included only in  $\mathbb{D}_T$  or in  $\mathbb{D}_V$ . In this way,  $\mathbb{D}_T$  and  $\mathbb{D}_V$  are completely disjoint sets (in terms of scenes), thus should lead to robust training.
2. *Fair-balanced- $N_T$* : training and validation shots are split according to scenes as for *Fair- $N_T$* . The number of shots for each device model is the same, leading to a model-balanced training dataset.
3. *Unfair- $P_T$* : training and validation shots are split regardless of the scene they belong to, fixing the percentage of training shots to  $P_T$ . In doing so, the same scene can appear in both training and validation sets, thus possibly leading to over-fitting and less accurate evaluation results.

A small case example for the three splitting strategies is available at Table 2.4.

## 2.5 Experimental Results

In this section we report the performed tests using different CNNs and training strategies to validate the proposed pipeline in a fair way.

Table 2.4.: A small scale example for three different splitting strategies. Row colors correspond to scenes. First, an instance id and a set of scenes are selected for the evaluation set  $\mathbb{D}_E$ . Considering the remaining instances and scenes,  $\mathbb{D}_T$  and  $\mathbb{D}_V$  are built according to what specified in the text. Labels E, V and T denote images associated to  $\mathbb{D}_E$ ,  $\mathbb{D}_V$  and  $\mathbb{D}_T$  according to each policy.

| Brand | Model   | Instance | <i>Fair</i> | <i>Fair-balanced</i> | <i>Unfair</i> |
|-------|---------|----------|-------------|----------------------|---------------|
| Canon | Ixus 70 | 0        | E           | E                    | E             |
| Canon | Ixus 70 | 0        |             |                      |               |
| Canon | Ixus 70 | 0        |             |                      |               |
| Canon | Ixus 70 | 1        |             |                      |               |
| Canon | Ixus 70 | 1        | V           | V                    | T             |
| Canon | Ixus 70 | 1        | T           | T                    | T             |
| Canon | Ixus 70 | 1        | T           |                      | V             |
| Kodak | M1063   | 0        | E           | E                    | E             |
| Kodak | M1063   | 0        |             |                      |               |
| Kodak | M1063   | 0        |             |                      |               |
| Kodak | M1063   | 1        |             |                      |               |
| Kodak | M1063   | 1        | V           | V                    | V             |
| Kodak | M1063   | 1        | T           | T                    | T             |

### 2.5.1 Impact Of The CNN Architecture

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

For each shot in the training, validation and evaluation sets in *Fair-60*,  $K = 32$  patches were extracted as described above. Training and validation patches were used to train the proposed CNNs. Specifically, the CNN architectures were trained on  $\mathbb{D}_T$  patches until classification loss on  $\mathbb{D}_V$  patches was minimized. Once the CNNs were trained, they were used to extract an 18 elements vector  $V_k$  for each patch  $P_k$  at the end of InnerProduct-2 layer

Table 2.5.: CNN classification accuracy for the different sets of *Fair-60*.

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

of the CNN. Results aggregation at shot level was performed averaging element by element feature vectors  $V_k$  associated to patches  $P_k$  belonging to the same shot  $P$ , so to obtain an 18 elements score vector  $V$  for the shot. Camera model associated to the maximum score was used to predict the shot’s class. Shots classification accuracy was computed on  $\mathbb{D}_T$ ,  $\mathbb{D}_V$  and  $\mathbb{D}_E$  as average over the 10 data realizations.

The results, presented in Table 2.5, indicate that the classification accuracy increases as we increase the CNN architecture depth: from 6 weight layers in the network  $\mathcal{M}_{Conv4}$  to 12 weight layers in the network  $\mathcal{M}_{Conv10}$ . The camera model classification accuracy of our architecture saturates when the depth reaches 12 layers, but even deeper CNNs might be beneficial for larger datasets with a higher number of classes and training images.

### 2.5.2 Impact Of Training Strategy

After testing different architectures, we focused on a reference CNN showing good performances and performed an extensive set of experiments over the three splitting policies described in the previous section. In particular, we selected the  $\mathcal{M}_{Conv4}$  CNN detailed in Table 2.2. For evaluation, we decided not to train additional SVMs, but to use the CNN output as class prediction. This allows us to study the effect of different training and validation split on the CNN only.

For this analysis, we fixed the number of evaluation scenes  $N_E$  to 10. For splitting policies *Fair* and *Fair-balanced* the number of training scenes  $N_T$  was varied in  $\{10, 15, 20, 30, 40, 50, 60\}$  over the 73 available scenes. The remaining  $73 - N_T$  scenes

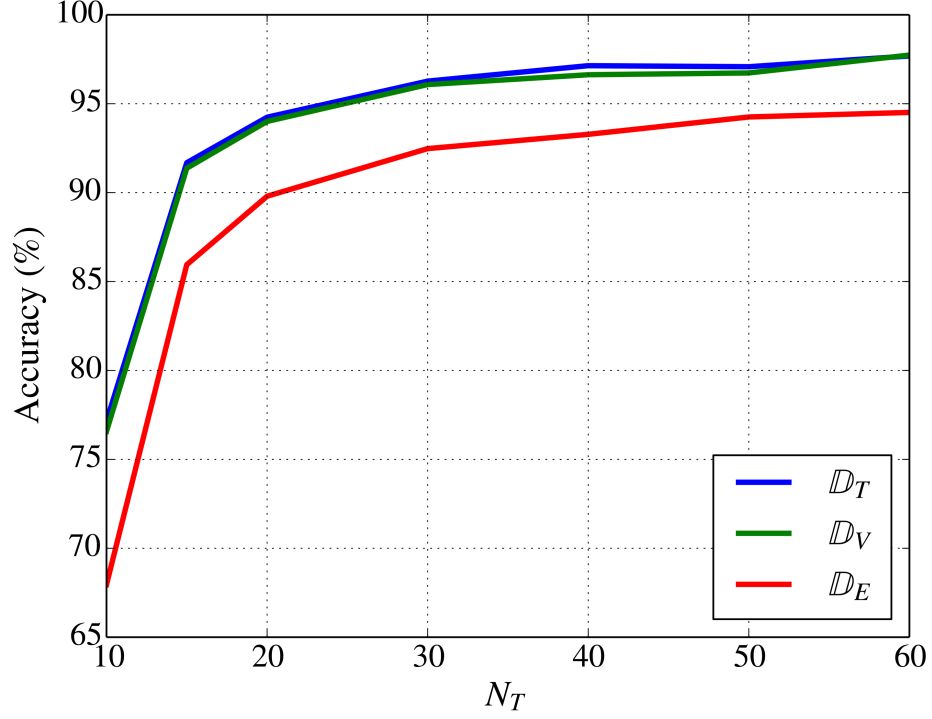


Fig. 2.4.: Fair splitting policy results. Training (blue), validation (green) and evaluation (red) set.

were used for validation. For splitting policy *Unfair* the percentage of training shots  $P_T$  was varied in  $\{10, 20, 30, 40, 50, 60, 70, 80, 90\}$ . The remaining shots were assigned to the validation set. This resulted in 23 splitting policies.

Figure 2.4 shows results using the *Fair* splitting policy to select train and validation datasets. Shots classification accuracy is reported as function of number of training scenes. Train and validation curves, in blue and green respectively, are almost always aligned. The red curve refers to the performance on the evaluation set. When using a small number of scenes for training (i.e.,  $N_t = 10$ ), the small amount of data limits the CNN capabilities of learning from data. Once the number of training scenes is sufficiently large (i.e.,  $N_t > 15$ ) the results increase reaching an evaluation accuracy up to 94.5%.

Figure 2.5 shows results using the *Fair-balanced* splitting policy to select train and validation datasets. In this case the small amount of data severely limits the CNN capabilities

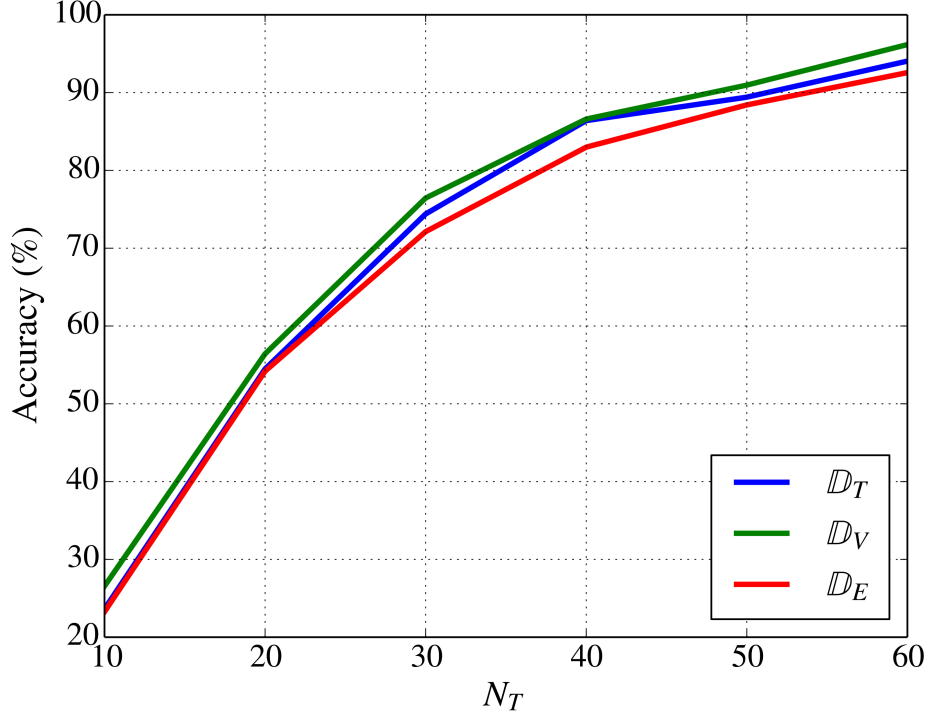


Fig. 2.5.: Fair balanced splitting policy results. Training (blue), validation (green) and evaluation (red) set.

of learning from data. In fact, in the Dresden Dataset, some camera models are represented by only a few number of shots. In the best situation (*Fair-balanced-60*), the evaluation accuracy reaches 92.6%.

Figure 2.6 shows results using the *Unfair* splitting policy to select train and validation datasets. Also in this case the small amount of training data impairs CNN learning capabilities when  $P_T = 0.1$ . However, as soon as the percentage of training data is increased, the evaluation accuracy reaches 94.4%.

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

A comparison between the *Fair* (Figure 2.4) and the *Unfair* (Figure 2.6) shows that there is not much gain in carefully splitting training and validation scenes. A possible



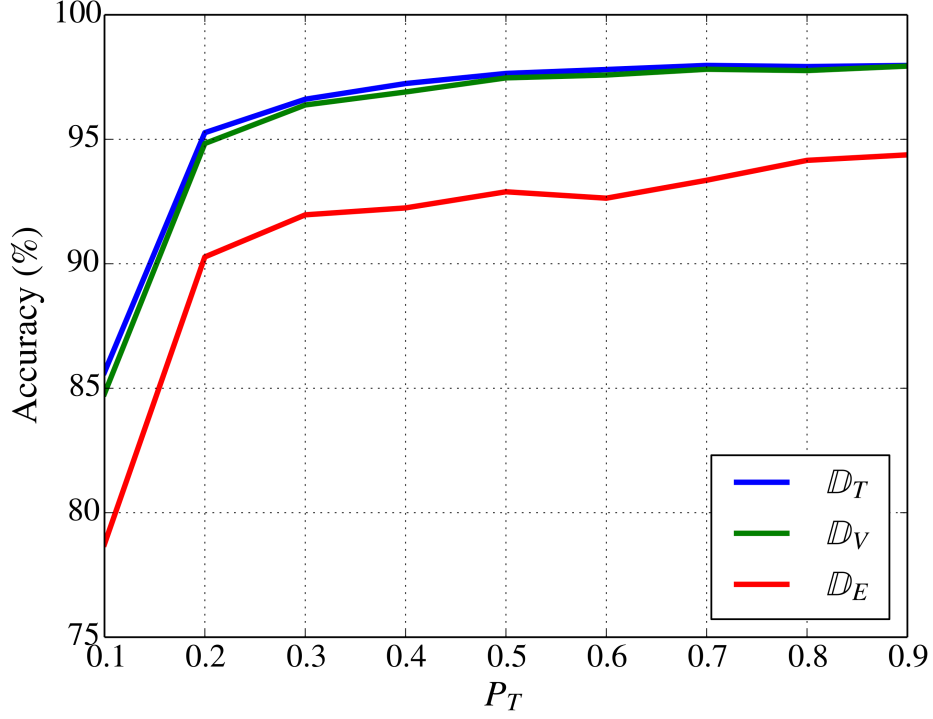


Fig. 2.6.: Unfair splitting policy results. Training (blue), validation (green) and evaluation (red) set.

motivation for this results stands in the small size of the patches used in this context. In fact, a  $64 \times 64 \times 3$  patch extracted from a full resolution picture (as the ones in the Dresden Image Dataset) contains only a few details from the image, and rarely some scene specific content that might be found only in larger patches. This motivates even further the use of small patches for this learning task.

### 2.5.3 Comparison With The State-Of-The-Art

After validating the performance of CNNs standalone (i.e., using the InnerProduct-2 output as score for each class), we focused on the evaluation of the entire pipeline (i.e., with SVMs and majority voting) in comparison with the recently proposed state-of-the-art method by Chen et al. [66]. In particular we stopped the forward step of  $\mathcal{M}_{Conv4}$  at the end

of the ReLU-1 layer in order to extract from each patch  $P_k$  a feature vector  $V_k$ . As dataset, we considered the *Fair-balanced-60* splitting policy.

Figure 2.7 shows how the average classification accuracy on shots from  $\mathbb{D}_E$  varies while increasing the number of voting patches for each image. The proposed CNN-based approach is depicted by the green line. Benchmark result using the approach proposed by Chen et al. [66] on  $64 \times 64$  color patches followed by majority voting is shown with the red line. As the method proposed by Chen et al. is not specifically tailored to small patches, we also tested it on full resolution images without voting procedures (i.e., blue line of Figure 2.7). It is worth noting that, despite the high accuracy obtained by Chen et al., our method approaches within 1% their result by using considerably less input data (i.e., just a few patches and not the full image).

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

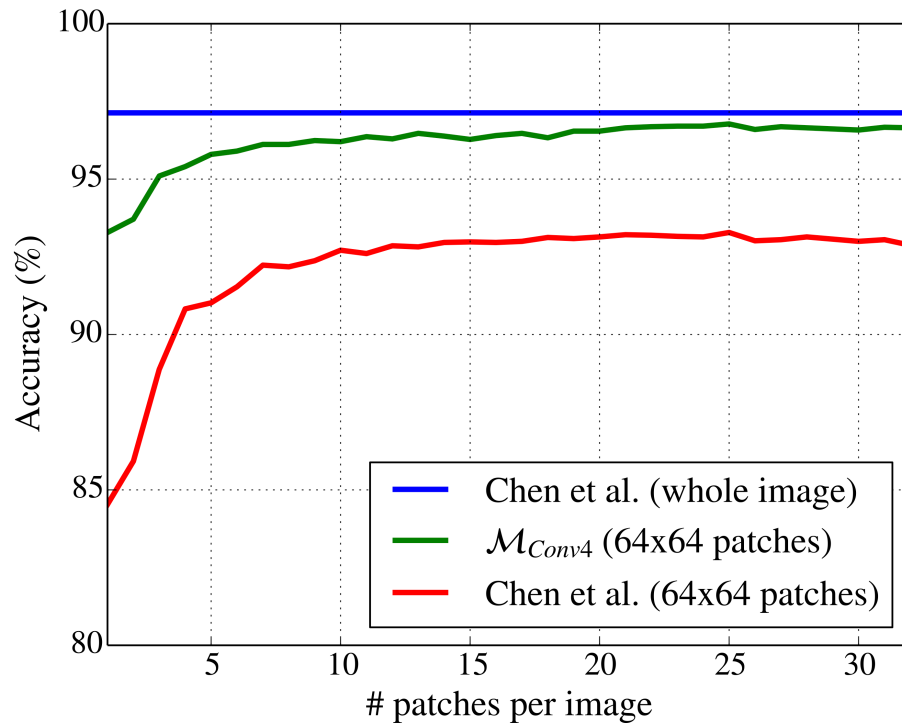


Fig. 2.7.: Comparison between the overall pipeline considering the CNN  $\mathcal{M}_{Conv4}$  trained with *Fair-balanced-60* splitting policy and the state-of-the-art algorithm by Chen et al. [66].

### 3. RELIABILITY MAP ESTIMATION FOR CAMERA IDENTIFICATION

#### 3.1 Overview

Due to the widespread availability of inexpensive image capturing devices (e.g., cameras and smartphones) and user-friendly editing software (e.g., GIMP and Adobe Photoshop), image manipulation is very easy. For this reason, the multimedia forensic community has developed techniques for image authenticity detection and integrity assessment [5, 54, 55].

Among the problems considered in the forensic literature, one important problem is camera model attribution, which consists in estimating the camera model used to acquire an image [58]. This proves useful when a forensic analyst needs to link an image under investigation to a user [57], or to detect possible image manipulations [59, 82] (e.g., splicing of pictures from different cameras).

Linking an image to a camera can in principle be trivially done exploiting image header information (e.g., EXIF data). It is also true that image headers are not reliable (e.g., anyone can tamper with them) or not always available (e.g., decoded images and screen captures). Therefore, the need for a series of blind methodologies has led to the development of pixel-based only information extraction methods.

These methods leverage the fact that image acquisition pipeline is slightly different for each camera model and manufacturer (e.g., different sensors and color equalization techniques). Therefore, each image contains characteristic “fingerprints” that enable one to understand which pipeline has been used and hence the camera model. Among these techniques, exploiting photo sensor non uniformity (PRNU) is particularly robust and enables camera instance identification [83, 84]. Other methods exploit traces left by color filter array (CFA) interpolation [61, 85, 86], camera lenses [60], histogram equalization [87]

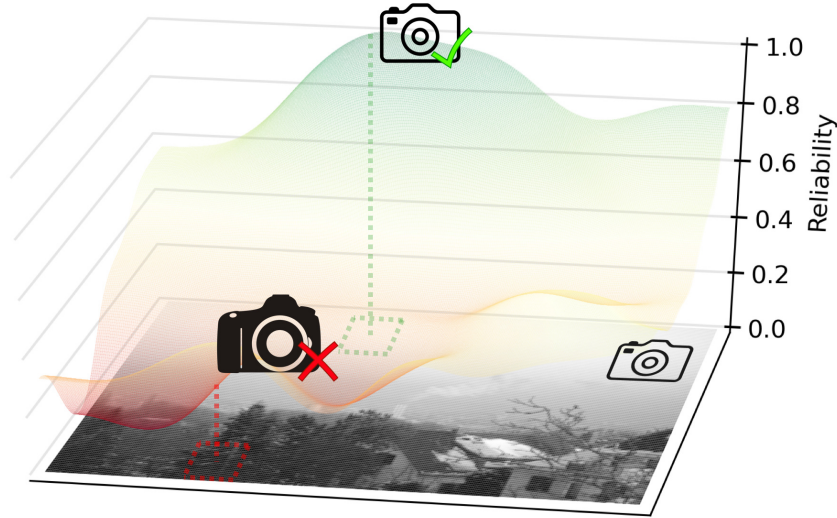


Fig. 3.1.: Reliability map representation for an example image taken with a given camera. In this case, patches belonging to the sky (green box) are more likely to provide accurate camera model attribution than patches containing textures (red box).

or noise [88]. Alternatively, a series of methods extracting statistical features from the pixel-domain and exploiting supervised machine-learning classifier have also been proposed [66–68].

Due to the advancements brought by deep learning techniques in the last few years, the forensic community is also exploring convolutional neural networks for camera model identification [89]. Interestingly, the approach in [6] has shown the possibility of accurately estimating the camera model used to acquire an image by analyzing a small portion of the image (i.e., a  $64 \times 64$  color image patch). This has lead to the development of forgery localization techniques [90].

In this chapter we propose a CNN-based method for estimating patch reliability for camera model attribution. As explained in [7], not all image patches contain enough discriminative information to estimate the camera model (e.g., saturated areas and too dark regions). Leveraging the network proposed in [6], we show how it is possible to determine whether an image patch contains reliable camera model traces for camera model attribution. Using this technique, we build a reliability map, which indicates the likelihood of each image region to be possibly used for camera model attribution, as shown in Figure 3.1. This

map can be used to select only reliable patches for camera model attribution. Additionally, it can also be used to drive tampering localization methods [90] by providing valuable information on which patches should be considered to be unreliable.

The proposed method leverages CNN feature learning capabilities and transfer learning training strategies. Specifically, we make use of a CNN composed by the architecture proposed in [6] as feature extractor, followed by a series of fully connected layers for patch reliability estimation. Transfer learning enables to preserve part of the CNN weights of [6], and train the whole architecture end-to-end with a reduced number of image patches. Our strategy is validated on the Dresden Image Database [81]. We first validate the proposed architecture and training strategy. Then, we compare the proposed solution against a set of baseline methodologies based on classic supervised machine-learning techniques. Finally, we show how it is possible to increase camera model attribution accuracy by more than 8% with respect to [6] using the proposed method.

## 3.2 Problem Statement And Related Work

In this section we introduce the problem formulation with the notation used throughout the chapter. We then provide the reader a brief overview about CNNs and their use in multimedia forensics.

### 3.2.1 Problem Formulation

Let us consider a color image  $\mathbf{I}$  acquired with camera model  $l$  belonging to a set of known camera models  $\mathcal{L}$ . In this chapter, we consider the patch-based closed-set camera model attribution problem as presented in [6]. Given an image  $\mathbf{I}$ , this means:

- Select a subset of  $K$  color patches  $\mathbf{P}_k$ ,  $k \in [1, K]$ .
- Obtain an estimate  $\hat{l}_k = \mathcal{C}(\mathbf{P}_k)$  of the camera model associated with each patch through a camera attribution function  $\mathcal{C}$ .

- Optionally obtain final camera model estimate  $\hat{l}$  through majority voting over  $\hat{l}_k$ ,  $k \in [1, K]$ .

Our goal is to detect whether a patch  $\mathbf{P}_k$  is a good candidate for camera model attribution estimation. To this purpose, we propose a CNN architecture that learns a function  $\mathcal{G}$  expressing the likelihood of a patch  $\mathbf{P}_k$  to provide correct camera model identification, i.e.,  $g_k = \mathcal{G}(\mathbf{P}_k)$ . High values of  $g_k$  indicate high probability of patch  $\mathbf{P}_k$  to provide correct camera information. Conversely, low  $g_k$  values are attributed to patches  $\mathbf{P}_k$  that cannot be correctly classified. Pixel-wise likelihood is then represented by means of a reliability map  $\mathbf{M}$ , showing which portion of an image is a good candidate to estimate image camera model, as shown in Figure 3.1.

### 3.2.2 Convolutional Neural Networks In Multimedia Forensics

In this section, we present a brief overview of the foundations of convolutional neural networks (CNNs) that are needed to follow the chapter. For a thorough review on CNNs, we refer the readers of this thesis to Chapter 9 of [16].

Deep learning and in particular CNNs have shown very good performance in several computer vision applications such as visual object recognition, object detection and many other domains such as drug discovery and genomics [20]. Inspired by how the human vision works, the layers of a convolutional network have neurons arranged in three dimensions, so each layer has a width height, and depth. The neurons in a convolutional layer are only connected to a small, local region of the preceding layer, so we avoid wasting resources as it is common in fully-connected neurons. The nodes of the network are organized in multiple stacked layers, each performing a simple operation on the input. The set of operations in a CNN typically comprises convolution, intensity normalization, non-linear activation and thresholding, and local pooling. By minimizing a cost function at the output of the last layer, the weights of the network are tuned so that they are able to capture patterns in the input data and extract distinctive features. CNNs enable learning data-driven, highly representative, layered hierarchical image features from sufficient training data

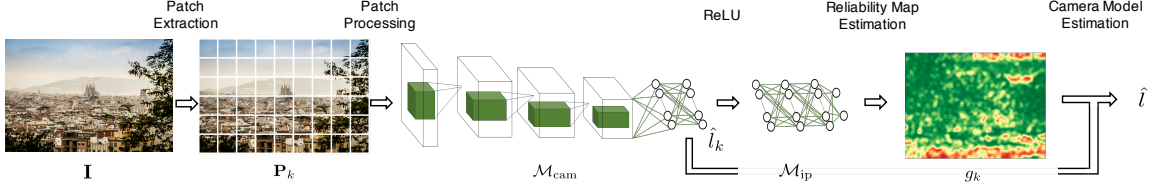


Fig. 3.2.: Block diagram of the proposed approach. Image  $I$  is split into patches. Each patch  $P_k$ ,  $k \in [0, K]$  is processed by the proposed CNN (composed by  $\mathcal{M}_{cam}$  and  $\mathcal{M}_{ip}$ ) to obtain a reliability score  $g_k$  and a camera model estimate  $\hat{l}_k$ . The reliability map is determined from all  $g_k$ ,  $k \in [0, K]$  values, and the overall picture camera model estimate  $\hat{l}$  can be computed.

There has been a growing interest in using convolutional neural networks in the fields of image forensics and steganalysis [73, 91]. These papers mainly focus on architectural design of CNNs where a single CNN model is trained and then tested in experiments. Data-driven models have recently proved valuable for other multimedia forensic applications as well [71, 72]. Moreover, initial exploratory solutions targeting camera model identification [6, 90, 92] show that it is possible to use CNNs to learn discriminant features directly from the observed known images, rather than having to use hand-crafted features. As a matter of fact, the use of CNNs also makes it possible to capture characteristic traces left by non-linear and hard to model operations present in the image acquisition pipeline of capturing devices.

In this thesis, we employ CNNs as base learners and test several different training strategies and network topologies. In our study, at first, a recently proposed CNN architecture is adopted as a feature extractor, trained on a random subsample of the training dataset. An intermediate feature representation is then extracted from the original data and pooled together to form new features ready for the second level of classification. Results have indicated that learning from intermediate representation in CNNs instead of output probabilities, and then jointly retraining the final architecture, leads to performance improvement.



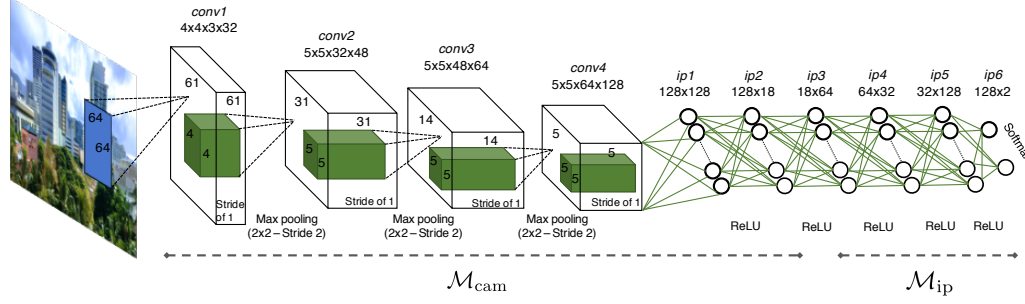


Fig. 3.3.: Representation of the proposed CNN architecture  $\mathcal{M}$  working on image patches. The first part ( $\mathcal{M}_{\text{cam}}$ ) computes a  $|\mathcal{L}|$ -element feature vector used for camera attribution. The second part ( $\mathcal{M}_{\text{ip}}$ ) is used to derive the camera model attribution reliability.

### 3.3 Patch Reliability Estimation Method<sup>1</sup>

In this section we provide details of our method for patch reliability estimation and camera attribution. The proposed pipeline is composed by the following steps (see Figure 3.2):

1. The image under analysis is split into patches
2. A CNN is used to estimate patch reliability likelihood
3. From the same CNN we estimate a camera model for each patch
4. A reliability mask is constructed and camera attribution of the whole image is performed

Below is a detailed explanation of each step.

#### 3.3.1 Patch Extraction

The proposed method works by analyzing image patches. The first step is to split the color image  $\mathbf{I}$  into a set of  $K$  patches  $\mathbf{P}_k$ ,  $k \in [0, K]$ . Each patch has  $64 \times 64$  pixel resolution. The patch extraction stride can range from 1 to 64 per dimension, depending on

<sup>1</sup>This is joint work with Mr. Sri Kalyan Yarlagadda, Prof. Fengqing Maggie Zhu, Prof. Paolo Bestagini, and Prof. Stefano Tubaro

the amount of desired overlap. This can be chosen to balance the trade-off between mask resolution reliability and computational burden.

### 3.3.2 Patch Camera Reliability

Each patch  $\mathbf{P}_k$  is input into the CNN  $\mathcal{M}$  shown in Figure 3.3, which can be logically split into two parts ( $\mathcal{M}_{\text{cam}}$  and  $\mathcal{M}_{\text{ip}}$ ) connected through a ReLU activation layer. Our proposed CNN learns a patch reliability function  $\mathcal{G}$  and returns the patch reliability  $g_k = \mathcal{G}(\mathbf{P}_k)$ .

The first part (i.e.,  $\mathcal{M}_{\text{cam}}$ ) is the CNN presented in [6] without last layer's activation. The rationale behind this choice is that this network is already known to be able to extract characteristic camera information. Therefore, we can mainly think of this portion of the proposed CNN as the feature extractor, turning a patch  $\mathbf{P}_k$  into a feature vector in  $\mathbb{R}^{|\mathcal{L}|}$ , where  $|\mathcal{L}|$  is the number of considered camera models. Formally,  $\mathcal{M}_{\text{cam}}$  is composed by:

- *conv1*: convolutional layer with 32 filters of size  $4 \times 4 \times 3$  and stride 1.
- *conv2*: convolutional layer with 48 filters of size  $5 \times 5 \times 32$  and stride 1.
- *conv3*: convolutional layer with 64 filters of size  $5 \times 5 \times 48$  and stride 1.
- *conv4*: convolutional layer with 128 filters of size  $5 \times 5 \times 64$  and stride 1, which outputs a vector with 128 elements.
- *ip1*: inner product layer with 128 output neurons followed by a ReLU layer to produce a 128 dimensional feature vector.
- *ip2*: final  $128 \times |\mathcal{L}|$  inner product layer.

The first three convolutional layers are followed by max-pooling layers with  $2 \times 2$  kernels and  $2 \times 2$  stride. This network contains 360 462 trainable parameters.

The second part of our architecture (i.e.,  $\mathcal{M}_{\text{ip}}$ ) is composed by a series of inner product layers followed by ReLU activations. This part of the proposed CNN can be considered as a two-class classifier trying to distinguish between patches that can be correctly classified,

and patches that cannot correctly be attributed to their camera model. As shall be clear in Section 3.4, we tested different possible  $\mathcal{M}_{\text{ip}}$  architecture candidates, to decide upon the following structure (denoted later on as  $\mathcal{M}_{\text{ip}}^4$  due to the 4 layers that characterize it):

- *ip3*: inner product layer with 64 output neurons followed by ReLU
- *ip4*: inner product layer with 32 output neurons followed by ReLU
- *ip5*: inner product layer with 128 output neurons followed by ReLU
- *ip6* inner product layer with 2 output neurons followed by softmax normalization

The first element of the softmax output vector can be considered the likelihood of a patch to be correctly classified. Therefore, we consider this value as  $g_k$ , and the transfer function learned by the whole  $\mathcal{M}$  network as  $\mathcal{G}$ .

### 3.3.3 Patch Camera Attribution

In order to detect the camera model from each patch  $\mathbf{P}_k$ , we exploit the architecture  $\mathcal{M}_{\text{cam}}$ . As explained in [6], this CNN output is a  $|\mathcal{L}|$ -element vector, whose argmax indicates the used camera model  $\hat{l}_k \in \mathcal{L}$ . Note that the softmax normalization at the end of  $\mathcal{M}_{\text{cam}}$  (as proposed in [6]) is not needed in this situation, as it only impacts the training strategy and not the argmax we are interested in.

### 3.3.4 Reliability-Map And Camera Attribution

In order to compute the reliability map  $\mathbf{M}$ , we aggregate all  $g_k$  values estimated from patches  $\mathbf{P}_k$ ,  $k \in [1, K]$ . This is done by generating a bidimensional matrix  $\mathbf{M}$  with the same size of image  $\mathbf{I}$ , and fill in the positions covered by the patch  $\mathbf{P}_k$  with the corresponding  $g_k$  values. In case of overlapping patches,  $g_k$  values are averaged. This map provides pixel-wise information about image regions from which reliable patches can be extracted. A few examples of estimated maps  $\mathbf{M}$  are reported in Figure 3.4.

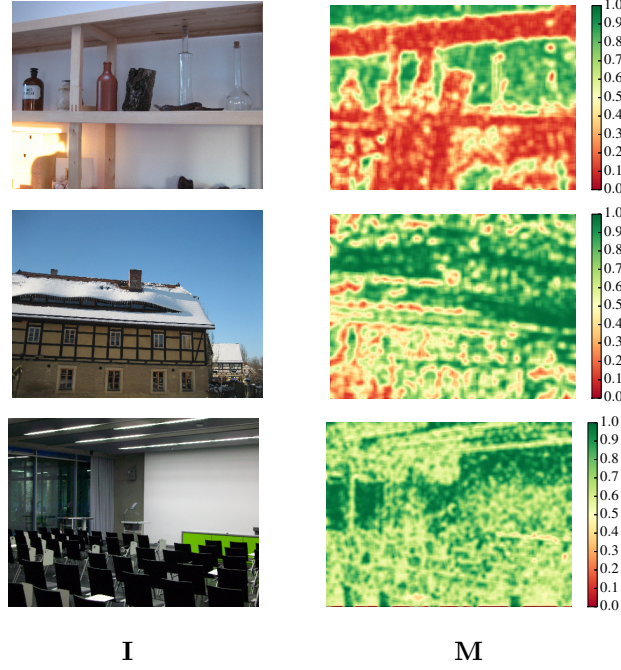


Fig. 3.4.: Examples of images **I** (left) and the estimated reliability maps **M** (right). Patch reliability is not always strictly linked to the image semantic content. Green areas represent high  $g_k$  values, thus reliable regions.

Finally, to attribute image **I** to a camera model, it is possible to select  $l_k$  only for the most reliable patch (i.e., highest  $g_k$ ), or perform majority voting on estimated  $l_k$  values belonging only to reliable regions, i.e.,  $\{l_k | g_k > \gamma\}$ , where  $\gamma \in [0, 1]$  is the reliability threshold (set experimentally to 0.5).

### 3.4 Experimental Results

In this section we report the details about our experiments. First, we describe the dataset. Then, we provide an insight on the used training strategies.

#### 3.4.1 Dataset

In this chapter we evaluate our solution adapting the dataset splitting strategy proposed in [6, 7, 58] to our problem. This strategy is tailored to the Dresden Image Dataset [81],

which consists of 73 devices belonging to 25 different camera models. A variable number of shots are taken with each device. Different motives are shot from each position. We refer to a scene as combination of geographical position with a specific motive. With this definition in mind, the dataset consists of a total of 83 scenes. Since we are trying to classify image patches at the level of camera model rather than instance level, we only consider camera models with more than one instance available. This leads us to a total of 18 camera models (as Nikon D70 and D70s basically differ only in their on-device screen) and nearly 15 000 shots.

In order to evaluate our method we divide the dataset consisting of 18 camera models into 3 sets, namely training  $\mathcal{D}_T$ , validation  $\mathcal{D}_V$  and evaluation  $\mathcal{D}_E$ .  $\mathcal{D}_T$  is again split into two equal sets  $\mathcal{D}_T^{\text{cam}}$  and  $\mathcal{D}_T^{\text{ip}}$ .  $\mathcal{D}_T^{\text{cam}}$  is used for training the parameters of  $\mathcal{M}_{\text{cam}}$ , whereas  $\mathcal{D}_T^{\text{ip}}$  is used for training  $\mathcal{M}_{\text{ip}}$ .  $\mathcal{D}_V$  is used to decide how many epochs to use during training to avoid overfitting. Finally,  $\mathcal{D}_E$  is used for evaluating the trained network on a disjoint set of images in a fair way.

While training a CNN, it is very important to avoid overfitting the data. In our dataset we have images of different scenes taken by different cameras, and our goal is to learn information about camera model from an image. As  $\mathcal{D}_V$  is used to avoid overfitting, it is important that  $\mathcal{D}_T$  and  $\mathcal{D}_V$  are sufficiently different from each other. It is also important that we test on data that has variation with respect to the training data. In order to achieve these goals, we do the following:

- Images for  $\mathcal{D}_E$  are selected from a single instance per camera and a set of 11 scenes.
- Images for  $\mathcal{D}_T$  are selected from the additional camera instances and 63 different scenes.
- Images in  $\mathcal{D}_V$  are selected from the same camera instances used for  $\mathcal{D}_T$ , but from the remaining 10 scenes. This makes sets  $\mathcal{D}_V$  and  $\mathcal{D}_T$  disjoint with respect to scenes, leading to robust training.

For training, validation and testing  $K = 300$  non overlapping color patches of size  $64 \times 64$  are extracted from each image. The resulting dataset  $\mathcal{D}_T^{\text{cam}}$  contains more than

500 000 patches split into 18 classes.  $\mathcal{D}_T^{\text{ip}}$  is reduced to 90 000 patches to balance reliable and non-reliable image patches according to  $\mathcal{M}_{\text{cam}}$  classification results. Finally,  $\mathcal{D}_V$  and  $\mathcal{D}_E$  are composed by more than 700 000 and 800 000 patches, respectively.

### 3.4.2 Training Strategies

Given that the proposed approach builds upon a pre-trained network (i.e.,  $\mathcal{M}_{\text{cam}}$ ), we propose a two-tiered transfer learning-based approach denoted as *Transfer*. For the sake of comparison, we also test two additional strategies, namely *Scratch* and *Pre-Trained*. In the following we report details about each strategy.

**Scratch.** This training strategy is the most simple one. It consists in training the whole two-class architecture  $\mathcal{M}$  using only  $\mathcal{D}_T^{\text{ip}}$  for training and  $\mathcal{D}_V$  for validation. This can be considered as a baseline training strategy. We use Adam optimizer with default parameters as suggested in [93] and batch size 128. Loss is set to binary-crossentropy.

**Pre-Trained.** This strategy takes advantage of the possibility of using a pre-trained  $\mathcal{M}_{\text{cam}}$ . In this case, we train  $\mathcal{M}_{\text{cam}}$  for camera model attribution using softmax normalization on its output. Training is carried out on  $\mathcal{D}_T^{\text{cam}}$  and validation on  $\mathcal{D}_V$ . Once  $\mathcal{M}_{\text{cam}}$  has been trained, we freeze its weights, and train the rest of the architecture  $\mathcal{M}_{\text{ip}}$  as a two-class classifier (i.e., reliable vs. non-reliable patches) using  $\mathcal{D}_T^{\text{ip}}$  and  $\mathcal{D}_V$ . Optimization during both training steps is carried out using Adam optimizer with default values and batches of 128 patches. We select categorical-crossentropy as our loss function.

**Transfer.** This two-tiered training strategy is meant to fully exploit the transfer learning capability of the proposed architecture. The first step consists in training  $\mathcal{M}_{\text{cam}}$  for camera model attribution using softmax normalization on its output and  $\mathcal{D}_T^{\text{cam}}$  and  $\mathcal{D}_V$  as datasets. This training step is optimized using Adam with default parameters, 128 patches per batch, and categorical-crossentropy as loss function.

The second step of  $\mathcal{M}$  training consists in freezing all the convolutional layers of  $\mathcal{M}_{\text{cam}}$ , and continue training all the inner product layers of both  $\mathcal{M}_{\text{cam}}$  and  $\mathcal{M}_{\text{ip}}$  using the datasets  $\mathcal{D}_T^{\text{ip}}$  and  $\mathcal{D}_V$ . This enables to jointly learn the weights of the classifier  $\mathcal{M}_{\text{ip}}$ , and tailor feature

extraction procedure in the last layers of  $\mathcal{M}_{\text{cam}}$  (i.e., *ip1* and *ip2*) to the classification task. For this step we use binary-crossentropy as loss, and stochastic gradient descent (SGD) with oscillating learning rate between  $5 \cdot 10^{-5}$  and  $15 \cdot 10^{-5}$  as optimizer. This choice is motivated by preliminary studies carried out in [94], and experimentally confirmed in our analysis.

### 3.5 Discussion

In this section we discuss the experimental results. First we show the capability of the proposed approach to distinguish between patches that contain camera model information and patches that are not suitable for this task. Then, we show how it is possible to improve camera model identification using the proposed approach.

#### 3.5.1 Patch Reliability

In order to validate the patch reliability estimation we perform a set of tests.

**CNN Architecture.** The first set of experiments has been devoted to the choice of a network architecture for  $\mathcal{M}_{\text{ip}}$ . To this purpose, we trained a set of different architectures for 15 epochs using the *Pre-Trained* strategy. As architectures we selected all possible combinations of up to six inner product layers (with ReLU activation) composed by 32, 64 or 128 neurons each. The last layer is always set to two neurons followed by softmax. From this experiment, we selected the model  $\mathcal{M}_{\text{ip}}$  with the highest validation accuracy for each tested amount of layers, which are:

- $\mathcal{M}_{\text{ip}}^2$  composed by two inner product layers with 128 and 2 neurons, respectively.
- $\mathcal{M}_{\text{ip}}^3$  composed by three inner product layers with 64, 128 and 2 neurons, respectively.
- $\mathcal{M}_{\text{ip}}^4$  composed by four inner product layers with 64, 32, 128 and 2 neurons, respectively.

- $\mathcal{M}_{\text{ip}}^5$  composed by five inner product layers with 64, 32, 64, 128 and 2 neurons, respectively.
- $\mathcal{M}_{\text{ip}}^6$  composed by six inner product layers with 64, 32, 32, 64, 64 and 2 neurons, respectively.

**Training Strategy.** In order to validate the proposed two-tiered *Transfer* training strategy, we trained the five selected models with the *Scratch*, *Pre-Trained* and *Transfer* strategies. Examples of training (on  $\mathcal{D}_{\text{T}}^{\text{ip}}$ ) and validation (on  $\mathcal{D}_{\text{V}}$ ) loss curves for the *Pre-Trained* and *Transfer* strategies on  $\mathcal{M}_{\text{ip}}^4$  are shown in Figure 3.5. It is possible to notice that the chosen optimizers enable a smooth loss decrease over several epochs. Moreover, the *Transfer* strategy provides a lower loss on both training and validation data, thus yielding better results compared to *Pre-Trained*. Similar conclusions can be drawn from the accuracy curves presented in Figure 3.5. We do not display curves for the *Scratch* strategy, as it is always worse than both the *Pre-Trained* and *Transfer* strategies. This was expected, as the amount of used training data in  $\mathcal{D}_{\text{T}}^{\text{ip}}$  is probably not enough to learn all parameters of  $\mathcal{M}$ . Therefore, starting from a pre-trained  $\mathcal{M}_{\text{cam}}$  becomes necessary.

Figure 3.6 shows the reliability patch estimation accuracy for all models from  $\mathcal{M}_{\text{ip}}^2$  to  $\mathcal{M}_{\text{ip}}^6$ , trained with all three training strategies and tested on the evaluation dataset  $\mathcal{D}_{\text{E}}$ . For each architecture, we selected the model with highest validation accuracy achieved over 100 epochs. These results further confirm that the *Scratch* strategy is not a viable solution for this problem. The *Transfer* strategy is the best choice for each network, achieving around 86% accuracy in detecting reliable patches. In other words, 86% of the selected patches will be correctly attributed to their camera, whereas only 14% of them will be wrongly classified. In the ideal scenario of errors uniformly spread across all models and images, this means we could use a majority voting strategy to further increase accuracy at the image level.

From Figure 3.6, it is also possible to notice that increasing  $\mathcal{M}_{\text{ip}}$  depth does not increase accuracy. Therefore, from this point on, we only consider architecture  $\mathcal{M}_{\text{ip}}^4$  as a good trade-off.



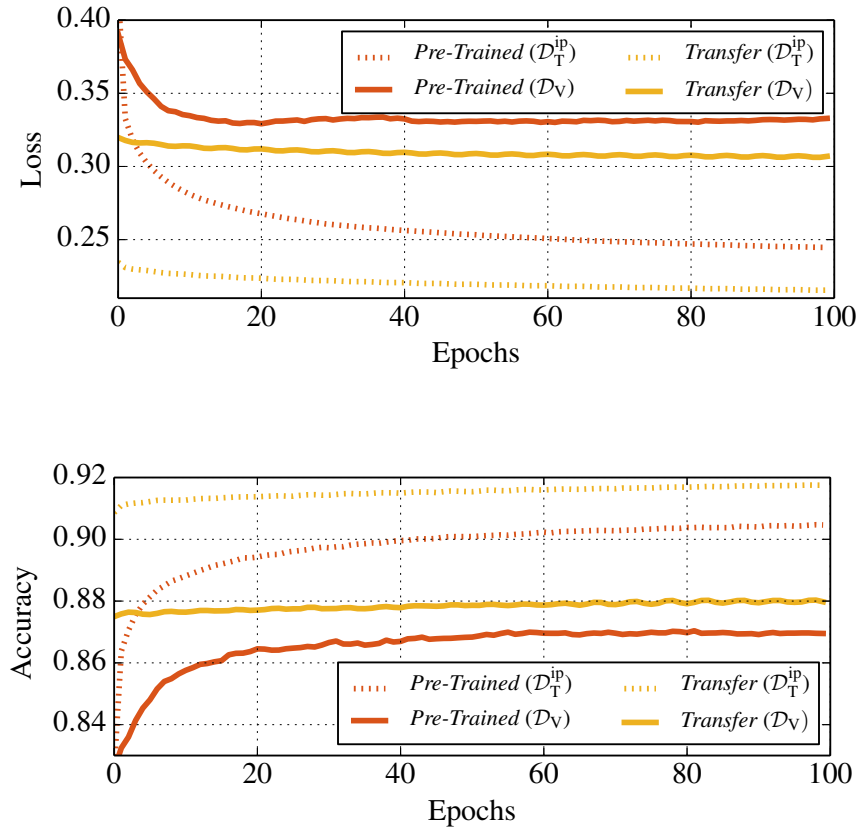


Fig. 3.5.: Loss and accuracy curves on training ( $\mathcal{D}_T^{ip}$ ) and validation ( $\mathcal{D}_V$ ) datasets using *Pre-Trained* and *Transfer* strategies on  $\mathcal{M}_{ip}^4$ .

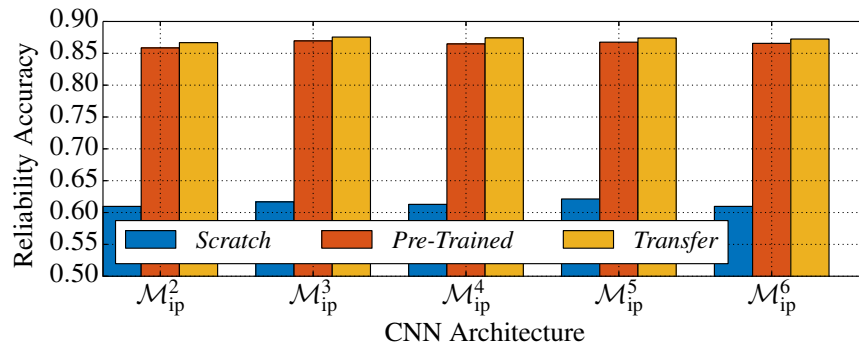


Fig. 3.6.: Patch reliability estimation accuracy. The *Transfer* strategy yields the most accurate results for every architecture.

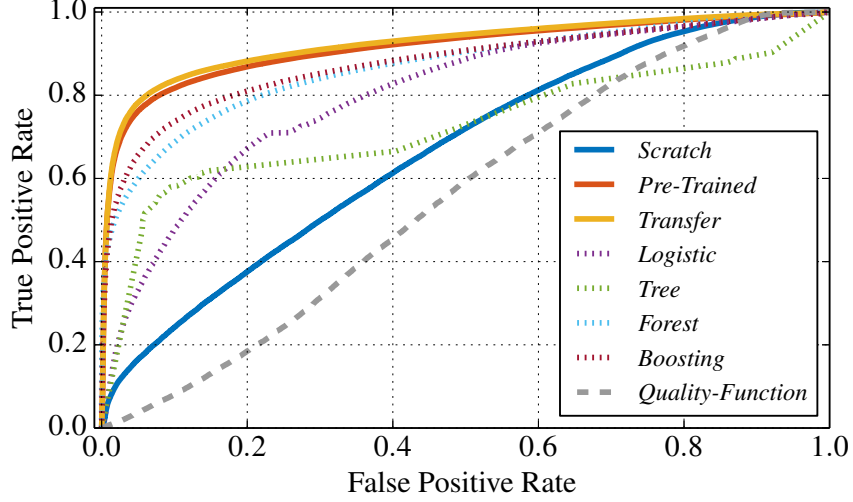


Fig. 3.7.: ROC curves on reliable patch detection. The proposed methods are represented by means of solid lines. Other baselines are dotted or dashed. Our *Transfer* strategy achieves the best overall performance.

**Baselines Comparison.** In order to further validate the proposed approach, we also considered two possible baseline solutions.

The first one consists in using other kinds of supervised classifiers exploiting the 18-element vector returned by  $\mathcal{M}_{\text{cam}}$  as feature. To this purpose, we trained a logistic regressor (*Logistic*), a decision tree (*Tree*), a random forest (*Forest*) and a gradient boosting classifier (*Boosting*). For each method, we applied z-score feature normalization and we selected the model achieving highest validation accuracy on  $\mathcal{D}_V$  after a parameter grid-search training on  $\mathcal{D}_T^{\text{ip}}$ . Accuracy results on patch reliability on evaluation set  $\mathcal{D}_E$  were 70.7%, 73.9%, 78.6% and 81.8%, respectively. None of them approaches the 86% of the proposed solution.

The second baseline solution we tested is the quality-function presented in [7] (*Quality-Function*). This function is computed for each patch and returns a value between zero and one indicating whether the patch is suitable for training  $\mathcal{M}_{\text{cam}}$ . Although Quality-Function was not intended to work as test reliability indicator, we decided that a comparison was necessary for completeness. To this purpose, Figure 3.7 shows receiver operating characteristic (ROC) curves obtained thresholding our reliability likelihood estimation  $g_k$ , the soft output of the other classifiers (i.e., logistic regressor, decision tree, etc.), and the quality-function

Table 3.1.: Camera model attribution accuracy using selected reliable patches from test dataset only. Using *Transfer* strategy (bold), the amount of selected patches in  $\mathcal{D}_E$  is always greater than 77% of  $|\mathcal{D}_E|$ . Accuracy improvement over random patch selection is greater than 8%.

| $\mathcal{M}_{ip}$   | Strategy           | Patches | Accuracy      | Acc. Delta    |
|----------------------|--------------------|---------|---------------|---------------|
| $\mathcal{M}_{ip}^2$ | <i>Scratch</i>     | 553 475 | 0.9009        | 0.0342        |
|                      | <i>Pre-Trained</i> | 618 958 | 0.9478        | 0.0811        |
|                      | <i>Transfer</i>    | 637 135 | <b>0.9513</b> | <b>0.0845</b> |
| $\mathcal{M}_{ip}^3$ | <i>Scratch</i>     | 518 228 | 0.9041        | 0.0374        |
|                      | <i>Pre-Trained</i> | 626 767 | 0.9520        | 0.0853        |
|                      | <i>Transfer</i>    | 641 808 | <b>0.9556</b> | <b>0.0888</b> |
| $\mathcal{M}_{ip}^4$ | <i>Scratch</i>     | 562 897 | 0.8963        | 0.0296        |
|                      | <i>Pre-Trained</i> | 649 515 | 0.9499        | 0.0832        |
|                      | <i>Transfer</i>    | 647 998 | <b>0.9532</b> | <b>0.0865</b> |
| $\mathcal{M}_{ip}^5$ | <i>Scratch</i>     | 511 425 | 0.9045        | 0.0378        |
|                      | <i>Pre-Trained</i> | 648 665 | 0.9529        | 0.0862        |
|                      | <i>Transfer</i>    | 651 508 | <b>0.9530</b> | <b>0.0863</b> |
| $\mathcal{M}_{ip}^6$ | <i>Scratch</i>     | 517 386 | 0.9035        | 0.0367        |
|                      | <i>Pre-Trained</i> | 651 405 | 0.9501        | 0.0834        |
|                      | <i>Transfer</i>    | 652 308 | <b>0.9531</b> | <b>0.0864</b> |

returned value [7]. As expected, the use of the quality-function presented in [7] provides less accurate results. Conversely, the proposed method achieves better performance than all other classifiers when trained according to *Transfer* strategy.

### 3.5.2 Camera Model Attribution

After validating the possibility of selecting reliable patches with the proposed method, we tested the effect of this solution on camera model attribution. To this purpose, we report in Table 3.1 the evaluation set results for the five investigated  $\mathcal{M}_{ip}$  models and the three training strategies when a single patch is used for camera model attribution. We do so in terms of:

1. *Patches*, i.e., the number of estimated reliable patches.
2. *Accuracy*, i.e., the average achieved camera model attribution result.

3. *Accuracy Delta*, i.e., the accuracy increment with respect to not using patch selection but randomly picking them (i.e., [6]).

These results highlight that it is possible to improve camera model attribution by more than 8%.

Figure 3.8 shows confusion matrix results using  $\mathcal{M}_{\text{cam}}$  (i.e., the output of the network proposed in [6]) on evaluation data  $\mathcal{D}_E$  randomly selecting patches. The average accuracy per patch is 87%. Figure 3.9 shows the same results, evaluating only patches considered reliable using  $\mathcal{M}_{\text{ip}}^4$ . In this scenario, accuracy increases to more than 95%. By comparing the two figures, it is possible to notice that many spurious classifications outside of the confusion matrix diagonal are corrected by the use of reliable patches only.

|            |             |                 |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|------------|-------------|-----------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| True label | Ixus70      | 0.90            | 0.01 |      |      |      |      | 0.01 | 0.03 |      |      | 0.03 | 0.01 |      |      |      |      |      |      |
|            | EX-Z150     |                 | 0.85 | 0.02 | 0.04 |      |      | 0.02 | 0.01 | 0.01 |      | 0.01 | 0.02 |      |      | 0.02 |      |      |      |
|            | FinePixJ50  |                 | 0.03 | 0.92 | 0.01 |      |      |      |      |      |      | 0.01 | 0.01 | 0.02 |      |      |      |      |      |
|            | M1063       |                 | 0.01 | 0.02 | 0.84 |      |      | 0.02 | 0.01 |      |      | 0.02 | 0.01 | 0.02 | 0.02 |      |      |      |      |
|            | CoolPixS710 |                 | 0.01 |      |      | 0.62 | 0.31 | 0.01 | 0.01 |      |      |      |      | 0.01 |      |      | 0.01 |      | 0.02 |
|            | D200        |                 |      |      |      | 0.13 | 0.83 |      |      |      |      |      |      | 0.01 | 0.01 |      | 0.01 |      | 0.01 |
|            | D70         |                 |      |      |      |      |      | 0.97 |      | 0.01 |      |      |      |      |      |      |      |      |      |
|            | mju-1050SW  |                 |      |      |      |      |      |      | 0.99 |      |      |      |      |      |      |      |      |      |      |
|            | DMC-FZ50    |                 | 0.01 |      |      |      |      | 0.01 | 0.01 | 0.93 |      | 0.01 | 0.01 |      |      |      |      |      |      |
|            | OptioA40    |                 |      |      |      |      |      | 0.01 |      |      | 0.94 | 0.03 |      |      | 0.01 |      |      | 0.01 |      |
|            | DCZ5.9      | 0.01            |      |      |      |      |      | 0.01 |      |      |      | 0.95 | 0.02 |      |      |      |      |      |      |
|            | GX100       |                 | 0.02 |      |      |      |      | 0.02 | 0.01 | 0.01 |      | 0.01 | 0.93 |      |      |      |      |      |      |
|            | RCP-7325XS  |                 | 0.01 |      | 0.04 |      |      |      |      |      |      |      | 0.01 | 0.90 |      | 0.02 |      |      | 0.01 |
|            | L74wide     |                 |      |      |      |      |      | 0.01 |      |      |      |      |      |      | 0.95 |      | 0.01 | 0.01 | 0.01 |
|            | NV15        |                 | 0.02 | 0.02 |      |      |      |      |      |      |      |      | 0.03 |      | 0.91 |      |      |      |      |
|            | DSC-H50     |                 |      |      |      |      |      | 0.01 |      |      |      |      |      |      |      | 0.65 | 0.01 | 0.31 |      |
|            | DSC-T77     |                 |      |      |      |      |      |      |      |      |      |      |      |      |      | 0.01 | 0.95 | 0.01 |      |
|            | DSC-W170    |                 |      |      |      |      |      | 0.01 |      |      |      |      |      | 0.01 |      | 0.39 | 0.01 | 0.57 |      |
|            | Ixus70      |                 |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|            | EX-Z150     |                 |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|            | FinePixJ50  |                 |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|            | M1063       |                 |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|            | CoolPixS710 |                 |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|            | D200        |                 |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|            | D70         |                 |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|            | mju-1050SW  |                 |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|            | DMC-FZ50    |                 |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|            | OptioA40    |                 |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|            | DCZ5.9      |                 |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|            | GX100       |                 |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|            | RCP-7325XS  |                 |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|            | L74wide     |                 |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|            | NV15        |                 |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|            | DSC-H50     |                 |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|            | DSC-T77     |                 |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|            | DSC-W170    |                 |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|            |             | Predicted label |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |

Fig. 3.8.: Camera model attribution confusion matrix obtained with  $\mathcal{M}_{\text{cam}}$  on  $\mathcal{D}_E$  without patch selection.

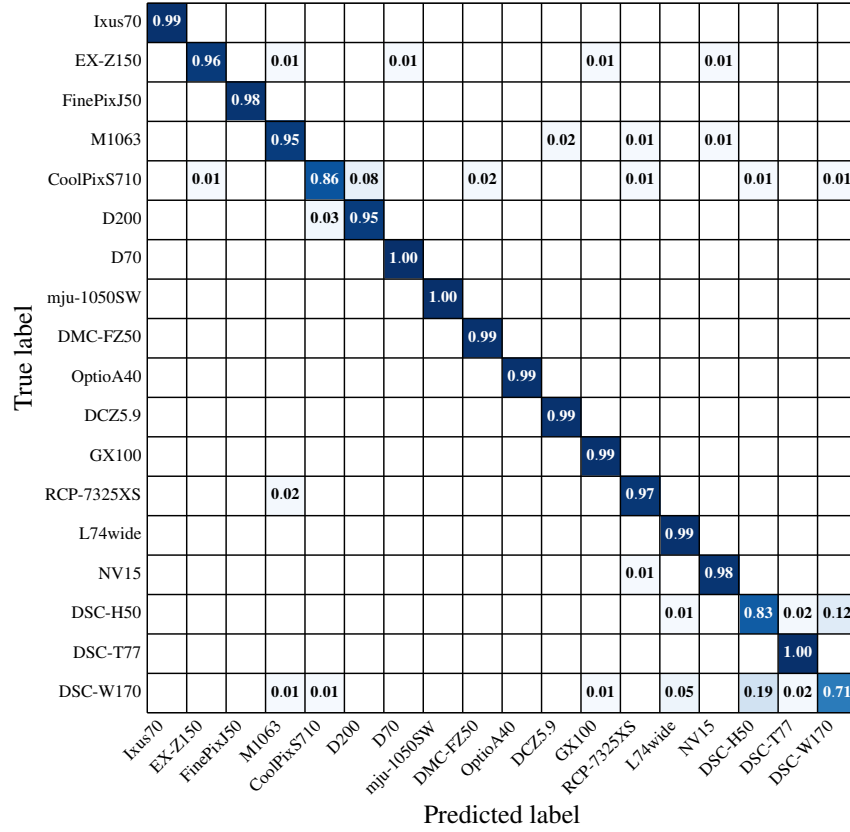


Fig. 3.9.: Camera model attribution confusion matrix obtained with  $\mathcal{M}_{\text{cam}}$  on  $\mathcal{D}_E$  using patches selected with  $\mathcal{M}_{\text{ip}}^4$ .

## 4. COUNTER-FORENSICS FOR CAMERA MODEL IDENTIFICATION

### 4.1 Overview

The recent increase in the number of digital images that are being uploaded and shared online has given rise to unique privacy and forensic challenges [95]. Among those challenges, verifying the integrity and authenticity of these widely circulated pictures is one of the most critical and complex tasks [5, 53].

In the last few years, the digital media forensic community has explored several techniques to evaluate the truthfulness of digital images and media [54, 55]. Due to its multiple applicable scenarios, research efforts have focused on camera model identification [56–58, 96, 97]. Determining the camera model used to take a picture can be very important in criminal investigations such as copyright infringement cases or where it is required to identify the authors of pedo-pornographic material.

Camera model identification can also be considered an important preliminary step to reduce the set of camera instances when we try to detect a unique camera instance rather than just the make and model [58]. In addition, being able to identify the camera model by inspecting small image regions is a viable method to uncover manipulation operations that could have been done to the image (e.g. splicing) [59].

Current camera model identification detectors make use of the fact that each camera model completes a distinctive set of tasks on each image when the device acquires the image. Examples of these tasks include the use of different JPEG compression schemes, application of proprietary methods for CFA demosaicing, and “defects” in the optical image path. Due to these characteristic operations, a singular “footprint” is embedded in each picture. This information can be utilized to identify the camera model, and perhaps the exact camera, that has been used to capture an image or record a video sequence.

Due to the inherent and growing complexity of the image acquisition pipeline of modern image capturing devices, it is a difficult challenge to adequately model the set of operations that a camera has to execute to capture an image. Successful attempts that use hand-crafted features to model the traces left by some of these operations can be found in [57, 60–63, 97, 98].

The use of deep learning techniques for image and video classification tasks [20, 99, 100] has shown that it is also possible to learn characteristic features that model a problem space directly from the data itself. This offers a viable path to leverage the growing amount of available image data. These modern approaches are data-driven in that they learn directly from the data rather than imposing a predetermined analytical model.

The data-driven model has recently proved valuable for forensics applications [70–73]. Initial exploratory solutions targeting camera model identification [6, 7, 92] show that it is possible to use CNNs to learn discriminant features directly from the observed known images, rather than having to use hand-crafted features. The use of CNNs also makes it possible to capture characteristic traces left by non-linear and hard to model operations present in the acquisition pipeline.

With the introduction of CNNs as detectors for camera model identification, a new vector for counter-forensic attacks is presented for a malevolent skilled individual. The idea of counter-forensics was first introduced in [101], where the authors presented the concept of fighting against image forensics with a practical application, namely a method for resampling an image without introducing pixel correlations. An up-to-date survey of the last counter-forensics advances can be found in [102].

Before exploring the vulnerabilities of CNN-based camera model detectors, it is important to note that detectors that rely on hand-crafted features are not immune to similar counter-forensics attacks. As explained in [103], digital camera fingerprints are vulnerable to forging. In particular, if an attacker obtains access to images from a given camera, they can estimate its fingerprint and “paste” it into an arbitrary image to make it look as if the image came from the camera with the stolen fingerprint. An early attempt to investigate such counter-forensic methods appeared in [104].

As presented in [105], several machine learning models, including state-of-the-art convolutional neural networks, are vulnerable to adversarial attacks. This means that these machine learning models misclassify images that are only slightly different from correctly classified images. In many cases, an ample collection of models with different architectures trained on different subsets of the training data misclassify the same adversarial example [106].

Although there are techniques such as adversarial training [105] or defensive distillation [107] that can slightly reduce the incidence of adversarial examples in CNN-based detectors, defending against adversarial examples is still an on-going challenge in the deep learning community. Adversarial attacks are hard to defend against because they require machine learning models that produce correct outputs for every possible input. The imposition of linear behavior when presented with inputs similar to the training data, though desirable, is precisely the main weakness of CNNs [106]. Due to the massive amount of possible inputs that a CNN can be presented with, it is remarkably simple to find adversarial examples that look unmodified to us but are misclassified by the network. Designing a truly adaptive defense against adversarial images remains an open problem.

In this chapter, we propose a counter-forensic method to subtly change an image to induce an error in its estimated camera model when analyzed by a CNN-based camera model detector. We leverage the recent developments to rapidly generate adversarial images. We test our counter-forensic method, using two well established adversarial image crafting techniques [106, 108], against an advanced deep learning architecture [24] carefully trained on a reference camera model dataset. Our results show that even modern and properly trained CNNs are susceptible to simple adversarial attacks. Note that our method only requires access to the predictions of the CNN-based camera model identification detector and does not need access to the weights of the CNN.



## 4.2 CNN-Based Camera Model Identification

In this section, we provide a brief overview of convolutional neural networks sufficient to understand the rest of this chapter and show how they can be used as camera model detectors. For a more detailed description, please refer to one of the several available tutorials in the literature [16, 109].

Convolutional neural networks are a special type of neural networks, biologically inspired by the human visual cortex system, that consist of a very high number of interconnected nodes, or neurons. The architecture of a CNN is designed to take advantage of the 2D structure of an input image. This is achieved with local connections and tied weights followed by some forms of pooling which results in translation invariant features. The nodes of the network are organized in multiple stacked layers, each performing a simple operation on the input.

The set of operations in a CNN typically comprises convolution, intensity normalization, non-linear activation and thresholding, and local pooling. By minimizing a cost function at the output of the last layer, the weights of the network are tuned so that they are able to capture patterns in the input data and extract distinctive features.

In a CNN, the features are learned using backpropagation [69] coupled with an optimization method such as gradient descent [110] and the use of large annotated training datasets. The shallower layers of the networks usually learn low-level visual features such as edges, simple shapes and color contrast, whereas deeper layers combine these features to identify complex visual patterns. Finally, fully-connected layers coupled with a softmax layer are commonly used to generate an output class label that minimizes the cost function.

For example, in the context of image classification, the last layer is composed of  $N$  nodes, where  $N$  is the number of classes, that define a probability distribution over the  $N$  visual category. The value of a given node  $p_i$ ,  $i = 1, \dots, N$  belonging to the last layer represents the probability of the input image to belong to the visual class  $c_i$ .

To train a CNN model for a specific image classification task we need to define the hyperparameters of the CNN, which range from the sequence of operations to be performed,

to the number of layers or the number and shape of the filters in convolutional layers. We must also define a proper cost function to be minimized during the training process. Finally, a dataset of training and test images, annotated with labels according to the specific task (e.g. camera models in our work) needs to be prepared.

Figure 4.1 shows an example of a CNN-based pipeline for camera model identification similar to the one presented in [6]. To train the CNN architecture, we use a given set of training and validation labeled image patches coming from  $N$  known camera models. For each color image  $I$ , associated to a specific camera model  $L$ ,  $K$  non-overlapping patches  $P_k$ ,  $k \in [1, K]$ , of size  $32 \times 32$  pixels are randomly extracted. Each patch  $P_k$  inherits the same label  $L$  of the source image. As trained CNN model  $\mathcal{M}$ , we select the one that provides the smallest loss on validation patches.

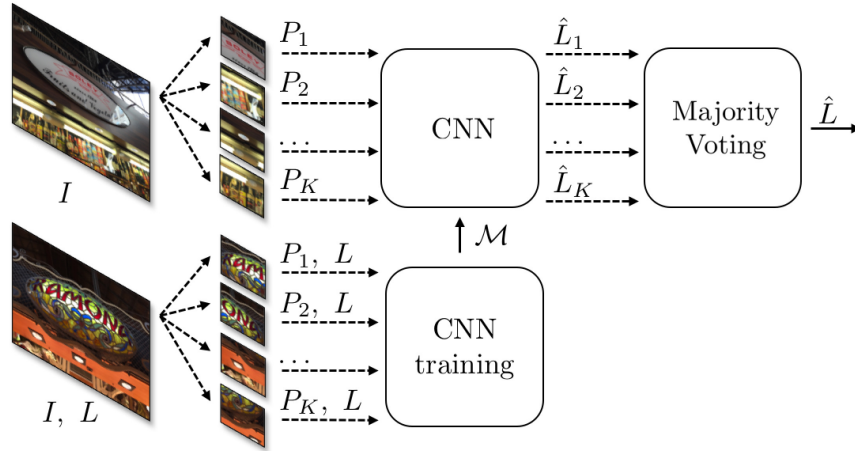


Fig. 4.1.: Example of a pipeline for camera model identification. The patches extracted from each training image  $I$  (bottom) inherit the same label  $L$  of the image. These patches are used in the CNN training process. For each patch  $P_k$  from the image  $I$  under analysis (top), a candidate label  $\hat{L}_k$  is produced by a trained CNN model  $\mathcal{M}$ . The predicted label  $\hat{L}$  for analyzed image  $I$  is obtained by majority voting.

When a new image  $I$  is under analysis, the camera model used to acquire it is estimated as follows. A set of  $K$  patches is obtained from image  $I$  as described above. Each patch  $P_k$  is processed by CNN model  $\mathcal{M}$  in order to assign a label  $\hat{L}_k$  to each patch. The predicted model  $\hat{L}$  for image  $I$  is obtained through majority voting on  $\hat{L}_k$ ,  $k \in [1, K]$ .

### 4.3 Proposed Method<sup>1</sup>

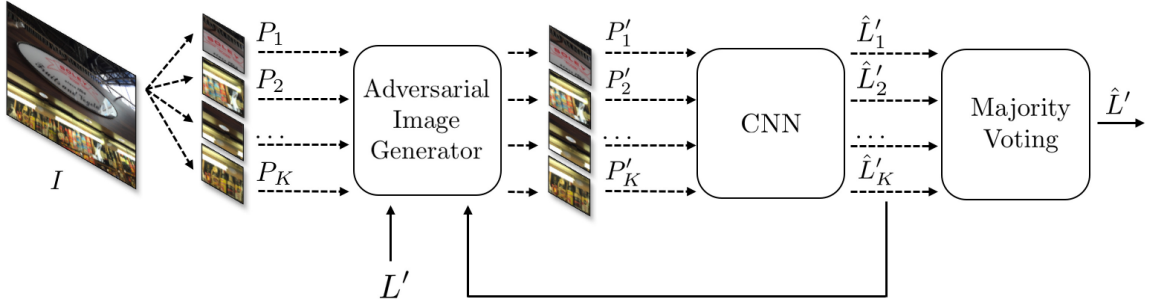


Fig. 4.2.: Block diagram of our proposed method.

Figure 4.2 shows the block diagram of our proposed counter-forensic method. Our method consists of an adversarial image generator module that can be added to a CNN-based camera model evaluation pipeline. In Figure 4.2, we assume a similar structure to the previously presented pipeline in Section 4.2. Our adversarial image generator module takes as input the set of  $K$  patches that have been extracted from the image  $I$  that is being analyzed. When presented with new image patches, our module can work in two different modes.

In the first operation mode, the adversarial image generator module does an untargeted image manipulation, that is, it does not try to perturb the image patches to produce a specific misclassification class. Instead, we use the derivative of the loss function of the CNN with respect to the input image patches to add a perturbation to the images. The derivative is computed using backpropagation with the labels  $\hat{L}'_k, k \in [1, K]$  that are given by the CNN detector when it first processes the unmodified image patches. This procedure is known as the fast gradient sign method (FGSM) [106].

In the second operation mode, the adversarial image generator module does a targeted image manipulation. In this case, we try to perturb the image patches to produce a specific misclassification class  $L'$ , different from the true real label  $L$  that is associated with the analyzed image  $I$  and its associated  $P_k$  patches. In this mode of operation, we exploit the

<sup>1</sup>This is joint work with Dr. Yu Wang, Dr. Luca Bondi, Prof. Paolo Bestagini, and Prof. Stefano Tubaro

forward derivative of a CNN to find an adversarial perturbation that will force the network to misclassify the image patch into the target class by computing the adversarial saliency map. Starting with an unmodified image patch, we perturb each feature by a constant offset  $\epsilon$ . This process is repeated iteratively until the target misclassification is achieved. This procedure is known as the Jacobian-based saliency map attack (JSMA) [108].

We present a detailed overview of both FGSM and JSMA techniques as follows.

#### 4.3.1 Fast Gradient Sign Method

In [106], the fast gradient sign method was introduced for generating adversarial examples using the derivative of the loss function of the CNN with respect to the input feature vector. Given an input feature vector (e.g. an image), FGSM perturbs each feature in the direction of the gradient by magnitude  $\epsilon$ , where  $\epsilon$  is a parameter that determines the perturbation size. For a network with loss  $J(\Theta, x, y)$ , where  $\Theta$  represents the CNN predictions for an input  $x$  and  $y$  is the correct label of  $x$ , the adversarial example is generated as

$$x^* = x + \epsilon \text{sign}(\nabla_x J(\Theta, x, y))$$

With small  $\epsilon$ , it is possible to generate adversarial images that are consistently misclassified by CNNs trained using the MNIST and CIFAR-10 image classification datasets with a high success rate [106].

#### 4.3.2 Jacobian-Based Saliency Map Attack

In [108], an iterative method for targeted misclassification was proposed. By exploiting the forward derivative of a CNN, it is possible to find an adversarial perturbation that will force the network to misclassify into a specific target class. For an input  $x$  and a convolutional neural network  $C$ , the output for class  $j$  is denoted  $C_j(x)$ . To achieve an output of target class  $t$ ,  $C_t(x)$  must be increased while the probabilities  $C_j(x)$  of all other

classes  $j \neq t$  decrease, until  $t = \arg \max_j C_j(x)$ . This is accomplished by exploiting the adversarial saliency map, which is defined as

$$S(x, t)[i] = \begin{cases} 0, & \text{if } \frac{\partial C_t(x)}{\partial x_i} < 0 \text{ or } \sum_{j \neq t} \frac{\partial C_j(x)}{\partial x_i} > 0 \\ \left( \frac{\partial C_t(x)}{\partial x_i} \right) / \left| \sum_{j \neq t} \frac{\partial C_j(x)}{\partial x_i} \right|, & \text{otherwise} \end{cases}$$

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

### 4.3.3 Implementation Details

To implement our counter-forensic method, we have used the software library *cleverhans* [111]. The library provides standardized reference implementations of adversarial image generation techniques and adversarial training. The library can be used to develop more robust CNN architectures and to provide standardized benchmarks of CNNs performance in an adversarial setting. As noted in [111], benchmarks constructed without a standardized implementation of adversarial image generation techniques are not comparable to each other, because a good result may indicate a robust CNN or it may merely indicate a weak implementation of the adversarial image generation procedure.

## 4.4 Experimental Results

In this section, we evaluate our proposed method and compare the results of the two techniques for generating the adversarial images. First, we create a reference dataset specially designed to exploit the traces left by the operations of the acquisition pipeline of

different image capturing devices. Then, we train an advanced deep learning architecture to have a baseline to compare the accuracy results in the presence of adversarial images. Finally, we generate several adversarial image examples to demonstrate the performance of our proposed method.

#### 4.4.1 Experimental Setup

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

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

Table 4.1 shows the statistics of the patch dataset. As can be seen, due to the difference in the number of images per camera model class in the PRNU-PAR dataset, our dataset of image patches has an unequal number of patches for each of the camera models.

Figure 4.3 shows a representative example of the images that are present in the PRNU-PAR dataset next to one of their randomly extracted patches. In this case, both camera models PAR-A075 and PAR-A106 have been used to capture images of a cloudy sky. Other camera models such AS-One or ES-D5100 have taken images of a white screen. All the image scenes that are captured in the PRNU-PAR dataset are mostly flat and bright.

Table 4.1.: Number of image patches per camera class for each of the different dataset splits.

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

As it has been shown in the literature [57], these largely uniform images are ideal candidates to be used for the extraction of the “fingerprint” (e.g. the characteristic PRNU noise of the camera model) left in the image by the camera.

#### 4.4.2 CNN Architecture

In order to do a fair evaluation of our counter-forensic method, we use a CNN-based camera model detector that has been trained to achieve state-of-the-art accuracy results in the patch dataset.

CNN architecture designs have tended to explore deeper models. Networks which can be hundreds of layers deep are now commonplace in the literature. This design trend has been motivated by the fact that for many applications such as image classification tasks, an increase in the depth of the CNN architecture translates into higher accuracy performance if sufficient amounts of training data are available.

A first approach to design a CNN architecture may be to simply stack convolutional or fully-connected layers together. This naive strategy works initially, but gains in accuracy performance quickly diminish the deeper this kind of architecture becomes. This



Fig. 4.3.: Example of images from the training set of the patch dataset. (Top) Image from camera model PAR-A075 and one of the randomly selected patches associated with it. (Bottom) Image from camera model PAR-A106 and one of the randomly selected patches associated with it.

phenomenon is due to the way in which conventional CNNs are trained through backpropagation. During the training phase of a CNN, gradient information must be propagated backwards through the network. This gradient information slightly diminishes as it passes through each layer of the neural network. For a CNN with a reduced number of layers, this is not a problem. For an architecture with a large number of layers, the gradient signal essentially becomes noise by the time it reaches the first layer of the network again.

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

A Residual Network [23], or ResNet is a deep CNN which tackles the problem of the vanishing gradient using a straightforward approach. It adds a direct connection at



each layer of the CNN. In previous CNN models, the gradient always has to go through the activations of the layers, which modify the gradient information due to the nonlinear activation functions that are commonly used. With this direct connection, the gradient could theoretically skip over all the intermediate layers and be propagated through the network without being disturbed.

A Dense Network [24], or DenseNet generalizes the idea of a direct connection between layers. Instead of only adding a connection from the previous layer to the next, it connects every layer to every other layer. For each layer, the feature maps of all preceding layers are treated as separate inputs whereas its own feature maps are passed on as inputs to all subsequent layers. The increased number of connections ensures that there is always a direct route for the information backwards through the network. The connectivity pattern of DenseNets yields state-of-the-art accuracies on the CIFAR10 image classification dataset, which is composed by images of  $32 \times 32$  pixels in size.

Motivated by the accuracy performance of DenseNet in the CIFAR10 dataset and the fact that we also work with image patches of  $32 \times 32$  pixels, we select a DenseNet model with 40 layers as our CNN camera model detector. To prevent the network from growing too wide and to improve the parameter efficiency, we limit the growth rate of the network, this is, the maximum number of input feature-maps that each layer can produce, to  $k = 12$ . To train the CNN, we use the Adam optimizer with a learning rate of 0.0001 and a batch size of 512 images. After 5 training epochs, we reach a plateau in the accuracy in our validation set. Table 4.2 shows the single patch accuracy results for our training, validation and test splits of the patch dataset.

Table 4.2.: Single patch accuracy results for our training, validation and test splits of the patch dataset.

| <b>Dataset Split</b> | <b>Train</b> | <b>Validation</b> | <b>Test</b> |
|----------------------|--------------|-------------------|-------------|
| Accuracy (%)         | 99.8         | 98.7              | 97.7        |

#### 4.4.3 Adversarial Image Generation

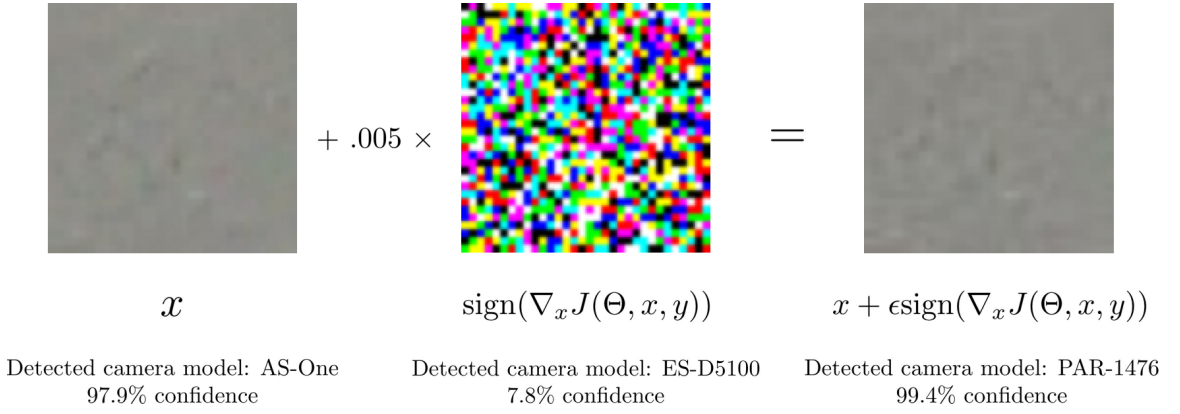


Fig. 4.4.: An example of untargeted fast adversarial image generation using FGSM applied to our trained DenseNet model on the patch dataset. By adding an imperceptibly small vector whose elements are equal to the sign of the elements of the gradient of the cost function with respect to the input, we can change DenseNet’s classification of the image patch.

In order to evaluate the performance of our counter-forensic method, we test the DenseNet model trained on the patch dataset using untargeted attacks with FGSM and targeted attacks with JSMA. To properly evaluate our method, we only perturb images from the test split which were correctly classified by our CNN in their original states.

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

For untargeted attacks with FGSM, we report in Table 4.3 the error rate and the average confidence score on the test split of the patch dataset for different values of  $\epsilon$  which have been shown to generate high misclassified adversarial images while not producing appreciable visual changes. We find that using  $\epsilon = 0.005$  offers the best compromise between error rate and visual changes in the image, causing the trained DenseNet model detector to

have a error rate of 93.1% with an average confidence of 95.3% on the patch test split. It should be noted that as we increase the value of  $\epsilon$ , the manipulations become more visually apparent.

Figure 4.4 shows an example of the adversarial images that our proposed method can generate when we use FGSM. The modifications done to the images by FGSM are performed on 32-bit floating point values, which are used for the input of the DenseNet model. The gradient computed for Figure 4.4 uses 8-bit signed integers. To publish the sign of the gradient image in the chapter, we have done a custom conversion from 8-bit signed integers to 8-bit unsigned integers. To increase the range of each color channel, we represent the  $-1$ s values as 0 and the 1s as 255. For the possible 0's, we have treated them as positive values (they are represented by 255).

Table 4.3.: Error rate and confidence score values of our trained DenseNet model after an untargeted attack with FGSM to the test split with different values of  $\epsilon$ .

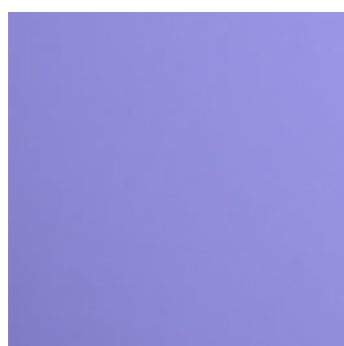
| $\epsilon$<br>value | Error<br>rate (%) | Confidence<br>Score (%) |
|---------------------|-------------------|-------------------------|
| 0.001               | 91.4              | 97.7                    |
| 0.002               | 91.7              | 97.2                    |
| 0.003             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

For targeted attacks with JSMA, we report in Table 4.4 the error rate and the average confidence score for each possible camera model target class. Figure 4.5 shows an example of the images that JSMA allows us to generate when we perform a targeted attack. In this case, an image patch captured by camera ES-D5100 that is correctly classified when is analyzed by our trained DenseNet model is manipulated to be misclassified as an image patch that had been generated by camera model PAR-1233. It is important to appreciate

Table 4.4.: Error rates and confidence scores of our trained DenseNet model for each possible target camera model after applying a targeted attack with JSMA to the test split.

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

that although JSMA allows us to generate image patches that get misclassified into a specific camera model with high error rates and confidence scores, the modifications that it applies to the images can usually be spotted through visual inspection. This effect is due to the fact that JSMA crafts the adversarial images by flipping pixels to their minimum or maximum values. Because our patch dataset is composed of image patches with mostly flat scene content, the effect can be clearly observed, for example, in the upper corners of the manipulated image patch in Figure 4.5.



Detected camera model: ES-D5100  
98.1% confidence



Detected camera model: PAR-1233  
89.6% confidence

Fig. 4.5.: An example of targeted adversarial image generation using JSMA applied to our trained DenseNet model on the patch dataset. (Left) Original image patch correctly classified as ES-D5100. (Right) Altered image patch with target camera model PAR-1233

## 5. DEEPPFAKE VIDEO DETECTION

### 5.1 Overview

The first known attempt at trying to swap someone’s face, circa 1865, can be found in one of the iconic portraits of U.S. President Abraham Lincoln. The lithography, as seen in Figure 5.1, mixes Lincoln’s head with the body of Southern politician John Calhoun. After Lincoln’s assassination, demand for lithographies of him was so great that engravings of his head on other bodies appeared almost overnight [112].

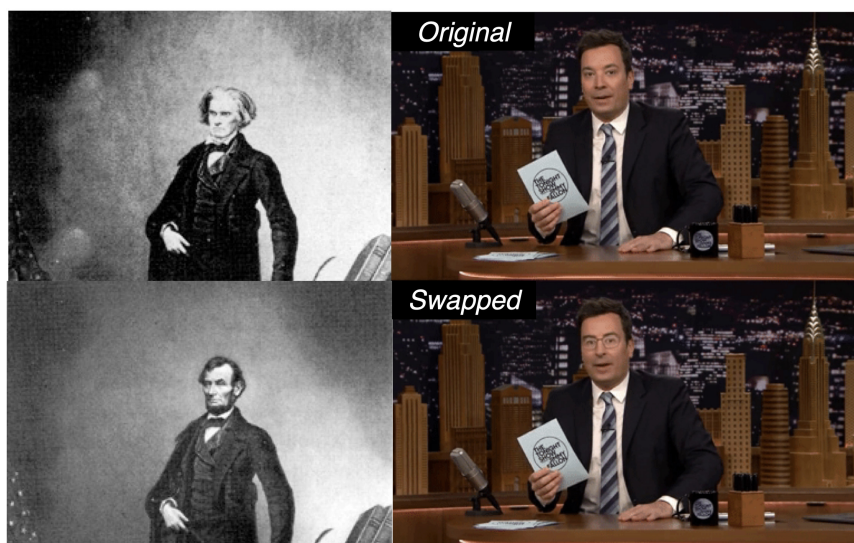


Fig. 5.1.: Face swapping is not new. Examples such as the swap of U.S. President Lincoln’s head with politician John Calhoun’s body were produced in mid-19th century (left). Modern tools like FakeApp [113] have made it easy for anyone to produce “deepfakes”, such as the one swapping the heads of late-night TV hosts Jimmy Fallon and John Oliver (right).

Recent advances [114, 115] have radically changed the playing field of image and video manipulation. The democratization of modern tools such as Tensorflow [116] or Keras [117] coupled with the open accessibility of the recent technical literature and cheap

access to compute infrastructure have propelled this paradigm shift. Convolutional autoencoders [39, 118] and generative adversarial network (GAN) [119, 120] models have made tampering images and videos, which used to be reserved to highly-trained professionals, a broadly accessible operation within reach of almost any individual with a computer. Smartphone and desktop applications like FaceApp [121] and FakeApp [113] are built upon this progress.

FaceApp automatically generates highly realistic transformations of faces in photographs. It allows one to change face hair style, gender, age and other attributes using a smartphone. FakeApp is a desktop application that allows one to create what are now known as “deep-fakes” videos. Deepfake videos are manipulated videoclips which were first created by a Reddit user, deepfake, who used TensorFlow, image search engines, social media websites and public video footage to insert someone else’s face onto pre-existing videos frame by frame.

Although some benign deepfake videos exist, they remain a minority. So far, the released tools [113] that generate deepfake videos have been broadly used to create fake celebrity pornographic videos or revenge porn [122]. This kind of pornography has already been banned by sites including Reddit, Twitter, and Pornhub. The realistic nature of deepfake videos also makes them a target for generation of pedopornographic material, fake news, fake surveillance videos, and malicious hoaxes. These fake videos have already been used to create political tensions and they are being taken into account by governmental entities [123].

As presented in the Malicious AI report [124], researchers in artificial intelligence should always reflect on the dual-use nature of their work, allowing misuse considerations to influence research priorities and norms. Given the severity of the malicious attack vectors that deepfakes have caused, in this chapter we present a novel solution for the detection of this kind of video.

The main contributions are summarized as follows. First, we propose a two-stage analysis composed of a CNN to extract features at the frame level followed by a temporally-aware RNN network to capture temporal inconsistencies between frames introduced by the

face-swapping process. Second, we have used a collection of 600 videos to evaluate the proposed method, with half of the videos being deepfakes collected from multiple video hosting websites. Third, we show experimentally the effectiveness of the described approach, which allows use to detect if a suspect video is a deepfake manipulation with 94% more accuracy than a random detector baseline in a balanced setting.

## 5.2 Related Work

**Digital Media Forensics.** The field of digital media forensics aims to develop technologies for the automated assessment of the integrity of an image or video. Both feature-based [125, 126] and CNN-based [8, 11] integrity analysis methods have been explored in the literature. For video-based digital forensics, the majority of the proposed solutions try to detect computationally cheap manipulations, such as dropped or duplicated frames [41] or copy-move manipulations [44]. Techniques that detect face-based manipulations include methods that distinguish computer generated faces from natural ones such as Conotter et al. [45] or Rahmouni et al. [127]. In biometry, Raghavendra et al. [46] recently proposed to detect morphed faces with two pre-trained deep CNNs and Zhou et al. [48] proposed detection of two different face swapping manipulations using a two-stream network. Of special interest to practitioners is a new dataset by Rössler et al. [128], which has about half a million edited images that have been generated with feature-based face editing [39].

**Face-based Video Manipulation Methods.** Multiple approaches that target face manipulations in video sequences have been proposed since the 1990s [37, 129]. Thies et al. demonstrated the first real-time expression transfer for faces. They later proposed Face2Face [39], a real-time facial reenactment system, capable of altering facial movements in different types of video streams. Alternatives to Face2Face have also been proposed [130].

Several face image synthesis techniques using deep learning have also been explored as surveyed by Lu et al. [131]. Generative adversarial networks (GANs) are used for aging alterations to faces [120], or to alter face attributes such as skin color [132]. Deep feature



interpolation [133] shows remarkable results in altering face attributes such as age, facial hair or mouth expressions. Similar results of attribute interpolations are achieved by Lampel et al. [134]. Most of these deep learning based image synthesis techniques suffer from low image resolution. Karras et al. [135] show high-quality synthesis of faces, improving the image quality using progressive GANs.

**Recurrent Neural Networks.** – Long Short Term Memory (LSTM) networks are a particular type of Recurrent Neural Network (RNN), first introduced by Hochreiter and Schmidhuber [136] to learn long-term dependencies in data sequences. When a deep learning architecture is equipped with a LSTM combined with a CNN, it is typically considered as “deep in space” and “deep in time” respectively, which can be seen as two distinct system modalities. CNNs have achieved massive success in visual recognition tasks, while LSTMs are widely used for long sequence processing problems. Because of the inherent properties (rich visual description, long-term temporal memory and end-to-end training) of a convolutional LSTM architecture, it has been thoroughly studied for other computer vision tasks involving sequences (e.g. activity recognition [137] or human re-identification in videos [138]) and has lead to significant improvements.

### 5.3 Deepfake Videos Exposed

Due to the way that FakeApp [113] generates the manipulated deepfake video, intra-frame inconsistencies and temporal inconsistencies between frames are created. These video anomalies can be exploited to detect if a video under analysis is a deepfake manipulation or not. Let us briefly explain how a deepfake video is generated to understand why these anomalies are introduced in the videos and how we can exploit them.

#### 5.3.1 Creating Deepfake Videos

It is well known that deep learning techniques have been successfully used to enhance the performance of image compression. Especially, the autoencoder has been applied for dimensionality reduction, compact representations of images, and generative models learn-

ing [139]. Thus, autoencoders are able to extract more compressed representations of images with a minimized loss function and are expected to achieve better compression performance than existing image compression standards. The compressed representations or latent vectors that current convolutional autoencoders learn are the first cornerstone behind the faceswapping capabilities of [113]. The second insight is the use of two sets of encoder-decoders with shared weights for the encoder networks. Figure 5.2 shows how these ideas are used in the training and generation phases that happen during the creation of a deepfake video.

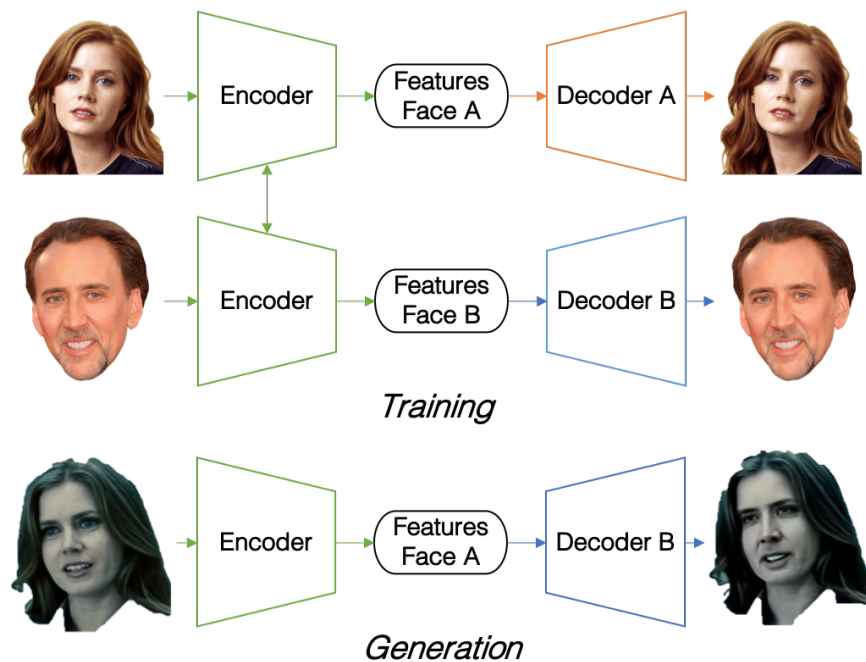


Fig. 5.2.: What makes deepfakes possible is finding a way to force both latent faces to be encoded on the same features. This is solved by having two networks sharing the same encoder, yet using two different decoders (top). When we want to do a new faceswapp, we encode the input face and decode it using the target face decoder (bottom).

## Training

Two sets of training images are required. The first set only has samples of the original face that will be replaced, which can be extracted from the target video that will be ma-

nipulated. This first set of images can be further extended with images from other sources for more realistic results. The second set of images contains the desired face that will be swapped in the target video. To ease the training process of the autoencoders, the easiest face swap would have both the original face and target face under similar viewing and illumination conditions. However, this is usually not the case. Multiple camera views, differences in lightning conditions or simply the use of different video codecs makes it difficult for autencoders to produce realistic faces under all conditions. This usually leads to swapped faces that are visually inconsistent with the rest of the scene. This frame-level scene inconsistency will be the first feature that we will exploit with our approach.

It is also important to note that if we train two autoencoders separately, they will be incompatible with each other. If two autoencoders are trained separately on different sets of faces, their latent spaces and representations will be different. This means that each decoder is only able to decode a single kind of latent representations which it has learnt during the training phase. This can be overcome by forcing the two set of autoencoders to share the weights for the encoder networks, yet using two different decoders. In this fashion, during the training phase these two networks are treated separately and each decoder is only trained with faces from one of the subjects. However, all latent faces are produced by the same encoder which forces the encoder itself to identify common features in both faces. This can be easily accomplished due to the natural set of shared traits of all human faces (e.g. number and position of eyes, nose, ...).

## **Video Generation**

When the training process is complete, we can pass a latent representation of a face generated from the original subject present in the video to the decoder network trained on faces of the subject we want to insert in the video. As shown in Figure 5.2, the decoder will try to reconstruct a face from the new subject, from the information relative to the original subject face present in the video. This process is repeated for every frame in the video where we want to do a faceswapping operation. It is important to point out that for

doing this frame-level operation, first a face detector is used to extract only the face region that will be passed to the trained autoencoder. This is usually a second source of scene inconsistency between the swapped face and the rest of the scene. Because the encoder is not aware of the skin or other scene information it is very common to have boundary effects due to a seamed fusion between the new face and the rest of the frame.

The third major weakness that we exploit is inherent to the generation process of the final video itself. Because the autoencoder is used frame-by-frame, it is completely unaware of any previous generated face that it may have created. This lack of temporal awareness is the source of multiple anomalies. The most prominent is an inconsistent choice of illuminants between scenes with frames, which leads to a flickering phenomenon in the face region common to the majority of fake videos. Although this phenomenon can be hard to appreciate to the naked eye in the best manually-tuned deepfake manipulations, it is easily captured by a pixel-level CNN feature extractor. The phenomenon of incorrect color constancy in CNN-generated videos is a well known and still open research problem in the computer vision field [140]. Hence, it is not surprising that an autoencoder trained with very constrained data fails to render illuminants correctly.

## 5.4 Recurrent Network For Deepfake Detection

In this section, we present our end-to-end trainable recurrent deepfake video detection system (Figure 5.3). The proposed system is composed by a convolutional LSTM structure for processing frame sequences. There are two essential components in a convolutional LSTM:

1. CNN for frame feature extraction.
2. LSTM for temporal sequence analysis.

Given an unseen test sequence, we obtain a set of features for each frame that are generated by the CNN. Afterwards, we concatenate the features of multiple consecutive frames and pass them to the LSTM for analysis. We finally produce an estimate of the likelihood of the sequence being either a deepfake or a nonmanipulated video.

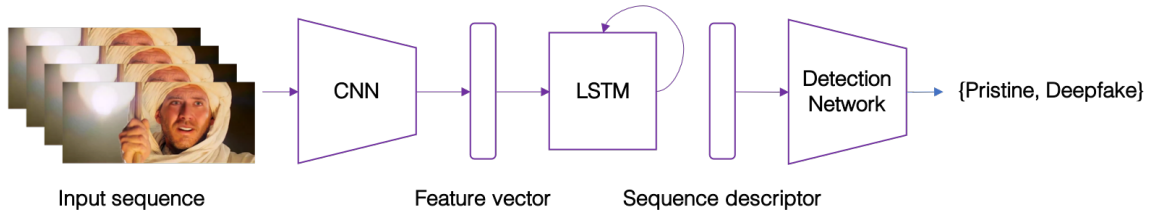


Fig. 5.3.: Overview of our detection system. The system learns and infers in an end-to-end manner and, given a video sequence, outputs a probability of it being a deepfake or a pristine video. It has a convolutional LSTM subnetwork, for processing the input temporal sequence.

### 5.4.1 Convolutional LSTM

Given an image sequence (see Figure 5.3), a convolutional LSTM is employed to produce a temporal sequence descriptor for image manipulation of the shot frame. Aiming at end-to-end learning, an integration of fully-connected layers is used to map the high-dimensional LSTM descriptor to a final detection probability. Specifically, our shallow network consists of two fully-connected layers and one dropout layer to minimize training over-fitting. The convolutional LSTM can be divided into a CNN and a LSTM, which we will describe separately in the following paragraphs.

**CNN for Feature Extraction.** Inspired by its success in the IEEE Signal Processing Society Camera Model Identification Challenge, we adopt the InceptionV3 [78] with the fully-connected layer at the top of the network removed to directly output a deep representation of each frame using the ImageNet pre-trained model. Following [141], we do not fine-tune the network. The 2048-dimensional feature vectors after the last pooling layers are then used as the sequential LSTM input.

**LSTM for Sequence Processing.** Let us assume a sequence of CNN feature vectors of input frames as input and a 2-node neural network with the probabilities of the sequence being part of a deepfake video or an untampered video. The key challenge that we need to address is the design of a model to recursively process a sequence in a meaningful manner. For this problem, we resort to the use of a 2048-wide LSTM unit with 0.5 chance of

dropout, which is capable to do exactly what we need. More particularly, during training, our LSTM model takes a sequence of 2048-dimensional ImageNet feature vectors. The LSTM is followed by a 512 fully-connected layer with 0.5 chance of dropout. Finally, we use a softmax layer to compute the probabilities of the frame sequence being either pristine or deepfake. Note that the LSTM module is an intermediate unit in our pipeline, which is trained entirely end-to-end without the need of auxiliary loss functions.

## 5.5 Experiments

In this section we report the details about our experiments. First, we describe our dataset. Then, we provide details of the experimental settings to ensure reproducibility and end up by analyzing the reported results.

### 5.5.1 Dataset

For this work, we have collected 300 deepfake videos from multiple video-hosting websites. We further incorporate 300 more videos randomly selected from the HOHA dataset [142], which leads to a final dataset with 600 videos. We selected the HOHA dataset as our source of pristine videos since it contains a realistic set of sequence samples from famous movies with an emphasis on human actions. Given that a considerable number of the deepfake videos are generated using clips from major films, using videos from the HOHA dataset further ensures that the overall system learns to spot manipulation features present in the deepfake videos, instead of memorizing semantic content from the two classes of videos present in the final dataset.

### 5.5.2 Parameter Settings

First, we have used a random 70/15/15 split to generate three disjoint sets, used for training, validation and test respectively. We do a balanced splitting, i.e., we do the splitting first for the 300 deepfake videos and then we repeat the process for the 300 nonmanipulated

videos. This guarantees that each final set has exactly 50% videos of each class, which allows use to report our results in terms of accuracy without having to take into account biases due to the appearance frequency of each class or the need of using regularizing terms during the training phase. In terms of data preprocessing of the video sequences, we do:

- Subtracting channel mean from each channel.
- Resizing of every frame to  $299 \times 299$ .
- Sub-sequence sampling of length  $N$  controlling the length of input sequence –  $N = 20, 40, 80$  frames. This allows use to see how many frames are necessary per video to have an accurate detection.
- The optimizer is set to Adam [93] for end-to-end training of the complete model with a learning rate of  $1e-5$  and decay of  $1e-6$ .

### 5.5.3 Results

It is not unusual to find deepfake videos where the manipulation is only present in a small portion of the video (i.e. the target face only appears briefly on the video, hence the deepfake manipulation is short in time). To account for this, for every video in the training, validation and test splits, we extract continuous subsequences of fixed frame length that serve as the input of our system.

In Table 5.1 we present the performance of our system in terms of detection accuracy using sub-sequences of length  $N = 20, 40, 80$  frames. These frame sequences are extracted sequentially (without frame skips) from each video. The entire pipeline is trained end-to-end until we reach a 10-epoch loss plateau in the validation set.

As we can observe in our results, with less than 2 seconds of video (40 frames for videos sampled at 24 frames per second) our system can accurately predict if the fragment being analyzed comes from a deepfake video or not with an accuracy greater than 97%.

Table 5.1.: Classification results of our dataset splits using video sub-sequences with different lengths.

| <b>Model</b>            | <b>Training<br/>acc. (%)</b> | <b>Validation<br/>acc. (%)</b> | <b>Test<br/>acc. (%)</b> |
|-------------------------|------------------------------|--------------------------------|--------------------------|
| Conv-LSTM,<br>20 frames | 99.5                         | 96.9                           | 96.7                     |
| Conv-LSTM,<br>40 frames | 99.3                         | 97.1                           | 97.1                     |
| Conv-LSTM,<br>80 frames | 99.7                         | 97.2                           | 97.1                     |

### FaceForensics++

To further verify the effectiveness of our approach, we have also tested it against the recently presented FaceForensics++ dataset by Rössler *et al.* [51]. This dataset is over an order of magnitude larger than comparable, publicly available, forgery datasets. Specifically, and thanks to recent contributions by researchers at Google AI [143], this dataset now has more than 4,000 DeepFake videos. Due to the larger size of this dataset, after careful exploration of the architectural space of our approach, we lower the number of units of the LSTM to 128, and the units of the fully connected layer to 32. Additionally, we also reduce the number of analyzed frames per sequence by the convolutional LSTM to 16. Furthermore, to ensure that no information leakage occurs due to difference in the way the pristine and manipulated videos are encoded, we reencode all the videos present in the FaceForensics++ dataset using the H.264 encoder, with a constant rate factor (CRF) of 23 and 30 frames per second (FPS). All these changes are primarily informed by hardware and software constraints.



Similar to the previously presented experiments, we have used a random 80/10/10 split to generate three disjoint sets, used for training, validation, and test respectively, while also ensuring they all have the same number of pristine and manipulated videos on them, and that they all cover a different set of scenes. Due to the larger amount and diversity of this dataset, our method now achieves a perfect (100%) accuracy detection rate in all three sets (note that accuracy is a valid metric in this case given the balanced number of positive and negative samples in the sets). We expect to further verify the validity of our approach by presenting it in the recently announced Deepfake Detection Challenge by Facebook [144, 145].


## **6. VIDEO MANIPULATION DETECTION USING STREAM DESCRIPTORS**

### **6.1 Overview**

Video manipulation is now within reach of any individual. Recent improvements in the machine learning field have enabled the creation of powerful video manipulation tools. Face2Face [39], Recycle-GAN [146], Deepfakes [147], and other face swapping techniques [148] embody the latest generation of these open source video forging methods. It is assumed as a certainty both by the research community [124] and governments across the globe [149, 150] that more complex tools will appear in the near future. Classical and current video editing methods have already demonstrated dangerous potential, having been used to generate political propaganda [151], revenge-porn [152], and child-exploitation material [153].

Due to the ever increasing sophistication of these techniques, uncovering manipulations in videos remains an open problem. Existing video manipulation detection solutions focus entirely on the observance of anomalies in the pixel domain of the video. Unfortunately, it can be easily seen from a game theoretic perspective that, if both manipulators and detectors are equally powerful, a Nash equilibrium will be reached [154]. Under that scenario, both real and manipulated videos will be indistinguishable from each other, and the best detector will only be capable of random guessing. Hence, methods that look beyond the pixel domain are critically needed. So far, little attention has been paid to the necessary metadata and auxiliary header information that is embedded in every video. As we shall present, this information can be exploited to uncover unskilled video content manipulators.

In this chapter, we introduce a new approach to address the video manipulation detection problem. To avoid the zero-sum, leader-follower game that characterizes current detection solutions, our approach completely avoids the pixel domain. Instead, we use the



|                                   |  |  |  |
|-----------------------------------|--|--|--|
| <b>video@<br/>codec_long_name</b> | H.264 / AVC / MPEG-4<br>AVC / MPEG-4 part 10 | H.264 / AVC / MPEG-4<br>AVC / MPEG-4 part 10 | H.264 / AVC / MPEG-4<br>AVC / MPEG-4 part 10 |
| <b>video@profile</b>              | Main   | High 4:4:4 Predictive                        | High   |
| <b>video@<br/>codec_time_base</b> | 0.02   | 0.016667                                     | 0.0208542                                    |
| <b>video@pix_fmt</b>              | yuv420p                                      | gbrp   | yuv420p                                      |
| <b>video@level</b>                | 31   | 50   | 40   |
| <b>video@bit_rate</b>             | 9.07642e+06                                  | 2.0245e+08                                   | 4.46692e+06                                  |

Fig. 6.1.: Examples of some of the information extracted from the video stream descriptors. These descriptors are necessary to decode and playback a video.

multimedia stream descriptors [155] that ensure the playback of any video (as shown in Figure 6.1). First, we construct a feature vector with all the descriptor information for a given video. Using a database of known manipulated videos, we train an ensemble of a support vector machine and a random forest that acts as our detector. Finally, during testing, we generate the feature vector from the stream descriptors of the video under analysis, feed it to the ensemble, and report a manipulation probability.

The contributions of this chapter are summarized as follows. First, we introduce a new technique that does not require access to the pixel content of the video, making it fast and scalable, even on consumer grade computing equipment. Instead, we rely on the multimedia descriptors present on any video, which are considerably harder to manipulate due to their role in the decoding phase. Second, we thoroughly test our approach using the NIST MFC datasets [156] and show that even with a limited amount of labeled videos, simple machine learning ensembles can be highly effective detectors. Finally, all of our code and trained classifiers will be made available<sup>1</sup> so the research community can reproduce our work with their own datasets.

<sup>1</sup><https://github.com/dguera/fake-video-detection-without-pixels>

## 6.2 Related Work

The multimedia forensics research community has a long history of trying to address the problem of detecting manipulations in video sequences. [157] provide an extensive and thorough overview of the main research directions and solutions that have been explored in the last decade. More recent work has focused on specific video manipulations, such as local tampering detection in video sequences [44, 158], video re-encoding detection [159, 160], splicing detection in videos [42, 161, 162], and near-duplicate video detection [163, 164]. [165, 166] also present solutions that use 3D PatchMatch [167] for video forgery detection and localization, whereas [168] suggest using data-driven machine learning based approaches. Solutions tailored to detecting the latest video manipulation techniques have also been recently presented. These include the works of [10, 49] on detecting Deepfakes and [128, 169] on Face2Face [39] manipulation detection.

As covered by [157], image-based forensics techniques that leverage camera noise residuals [170], image compression artifacts [171], or geometric and physics inconsistencies in the scene [172] can also be used in videos when applied frame by frame. In [173] and [174], Exif image metadata is used to detect either image brightness and contrast adjustments, and splicing manipulations in images, respectively. Finally, [175] use video file container metadata for video integrity verification and source device identification. To the best of our knowledge, video manipulation detection techniques that exploit the multimedia stream descriptors have not been previously proposed.

## 6.3 Proposed Method<sup>2</sup>

Current video manipulation detection approaches rely on uncovering manipulations by studying pixel domain anomalies. Instead, we propose to use the multimedia stream descriptors of videos as our main source of information to spot manipulated content. To do so, our method works in two stages, as presented in Figure ?? . First, during the training phase, we extract the multimedia stream descriptors from a labeled database of manipulated

---

<sup>2</sup>This is joint work with Mr. Sriram Baireddy, Prof. Paolo Bestagini, and Prof. Stefano Tubaro

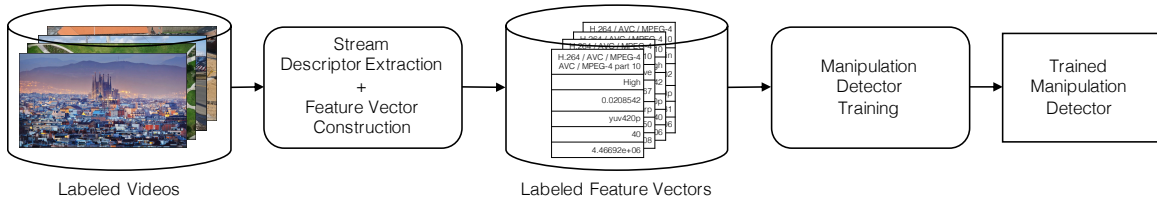


Fig. 6.2.: Block diagram of the training stage of our proposed method. We process a labeled database of manipulated and pristine videos to generate a feature vector for each video from its multimedia stream descriptors. These feature vectors are then used to train and select the best detector.

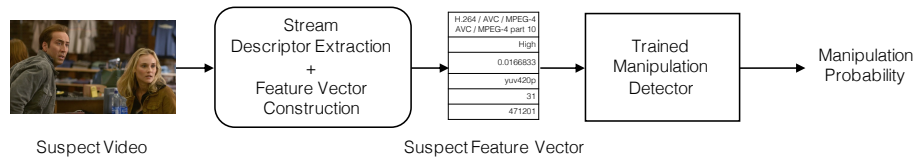


Fig. 6.3.: Block diagram of the testing stage of our proposed method. Given a suspect video, a feature vector is generated and processed by the previously selected detector. Finally, a manipulation probability for the suspect video is reported.

and pristine videos. In practice, such a database can be easily constructed using a limited amount of manually labeled data coupled with a semi-supervised learning approach, as done by [176]. Then, we encode these descriptors as a feature vector for each given video. We apply median normalization to all numerical features. As for categorical features, each is encoded as its own unique numerical value. Once we have processed all the videos in the database, we use all the feature vectors to train different binary classifiers as our detectors. More specifically, we use a random forest, a support vector machine (SVM) and an ensemble of both detectors. The best hyperparameters for each detector are selected by performing a random search cross-validation over a 10-split stratified shuffling of the data and 1,000 trials per split. Figure 6.2 summarizes this first stage. In our implementation, we use *ffprobe* [177] for the multimedia stream descriptor extraction. For the encoding of the descriptors as feature vectors, we use *pandas* [178] and *scikit-learn* [179]. As for the training and testing of the SVM, the random forest, and the ensemble, we use the implementations available in the *scikit-learn* library.

Figure 6.3 shows how our method would work in practice. Given a suspect video, we extract its stream descriptors and generate its corresponding feature vector, which is normalized based on the values learnt during the training phase. Since some of the descriptor fields are optional, we perform additional post-processing to ensure that the feature vector can be processed by our trained detector. Concretely, if any field is missing in the video stream descriptors, we perform data imputation by mapping missing fields to a fixed numerical value. If previously unseen descriptor fields are present in the suspect video stream, they are ignored and not included in the corresponding suspect feature vector. Finally, the trained detector analyzes the suspect feature vector and computes a manipulation probability.

It is important to note that although our approach may be vulnerable to video re-encoding attacks, this is traded off for scalability, a limited need of labeled data, and a high video manipulation detection score, as we present in Section 6.4. Also, the fact that our solution is orthogonal to pixel-based methods and requires limited amounts of data, which means that ideally, we could use both approaches simultaneously. Our approach could be used to quickly identify manipulated videos, minimizing the need to rely on human annotation. Later, these newly labeled videos could be used to improve the performance of pixel-based video manipulation detectors. Finally, following the recommendations of [124], we want to reflect on a potential misuse of the proposed approach. We believe that our approach could be misused by someone with access to large amounts of labeled video data. Using that information, a malevolent adversary could identify specific individuals, such as journalists or confidential informants, who may submit anonymous videos using the same devices they use to upload videos to social media websites. To avoid this, different physical devices or proper video data sanitization should be used.

## 6.4 Experimental Results

### 6.4.1 Datasets

In order to evaluate the performance of our proposed approach, we use the Media Forensics Challenge (MFC) datasets [156]. Collected by the National Institute of Standards and Technology (NIST), this data comprises over 11,000 high provenance videos and 4,000 manipulated videos. In our experiments, we use the videos from the following datasets for training, hyper-parameter selection, and validation: the Nimble Challenge 2017 development dataset, the MFC18 development version 1 and version 2 datasets, and the MFC18 GAN dataset. This represents a total of 677 videos, of which 167 are manipulated. For testing our model, we use the MFC18 evaluation dataset and the MFC19 validation dataset, which have a total of 1,097 videos. Of those videos, 336 have been manipulated.

### 6.4.2 Experimental Setup

To show the merits of our method in terms of scalability and limited compute requirements, we design the following experiment. First, we select machine learning binary classifiers that are well known for their modeling capabilities, even with limited access to training samples. As previously mentioned, we use a random forest, a support vector machine, and a soft voting classification ensemble with both. This final ensemble is weighted 4 to 1 in favor of the decision of the random forest. Then, to show the performance of each detector under different data availability scenarios, we train them using 10%, 25%, 50% and 75% of the available training data. We use a stratified shuffle splitting policy to select these training subsets, meaning that the global ratio of manipulated to non-manipulated videos of the entire training set is preserved in the subsets. In all scenarios, a sequestered 25% subset of the training data is used for hyper-parameter selection and validation. Finally, the best validated model is selected for testing. Due to the imbalance of manipulated to non-manipulated videos, we use the Precision-Recall (PR) curve as our evaluation metric,

as recommended by [180]. We also report the F1 score, the area under the curve (AUC) score, and the average precision (AP) score for each classifier.

### 6.4.3 Results and Discussion

As we can see in Figure 6.4, Figure 6.5, Figure 6.6, and Figure 6.7, under all scenarios the voting ensemble of the random forest and the support vector machine generally achieves the best overall results, followed by the random forest and the SVM. More specifically, our best ensemble model achieves a F1 score of 0.917, an AUC score of 0.984 and an AP score of 0.984. To contextualize these results, we have included the performance of a binary classifier baseline which predicts a video manipulation with probability  $p = 0.306$ . This corresponds to the true fraction of manipulated videos in the test set. Note that it is higher than the fraction of manipulated videos in the training subsets, which is 0.247. This baseline model would achieve an F1, AUC, and AP score of 0.306. We can see that our best model is three times better than the baseline in all reported metrics. Notice that, as seen in Figure 6.4, the ensemble trained with 68 videos has achieved equal or better results than the ensembles trained with more videos. This shows that, even with a very limited number of stream descriptors, a properly tuned machine learning model can be trained easily to spot video manipulations.



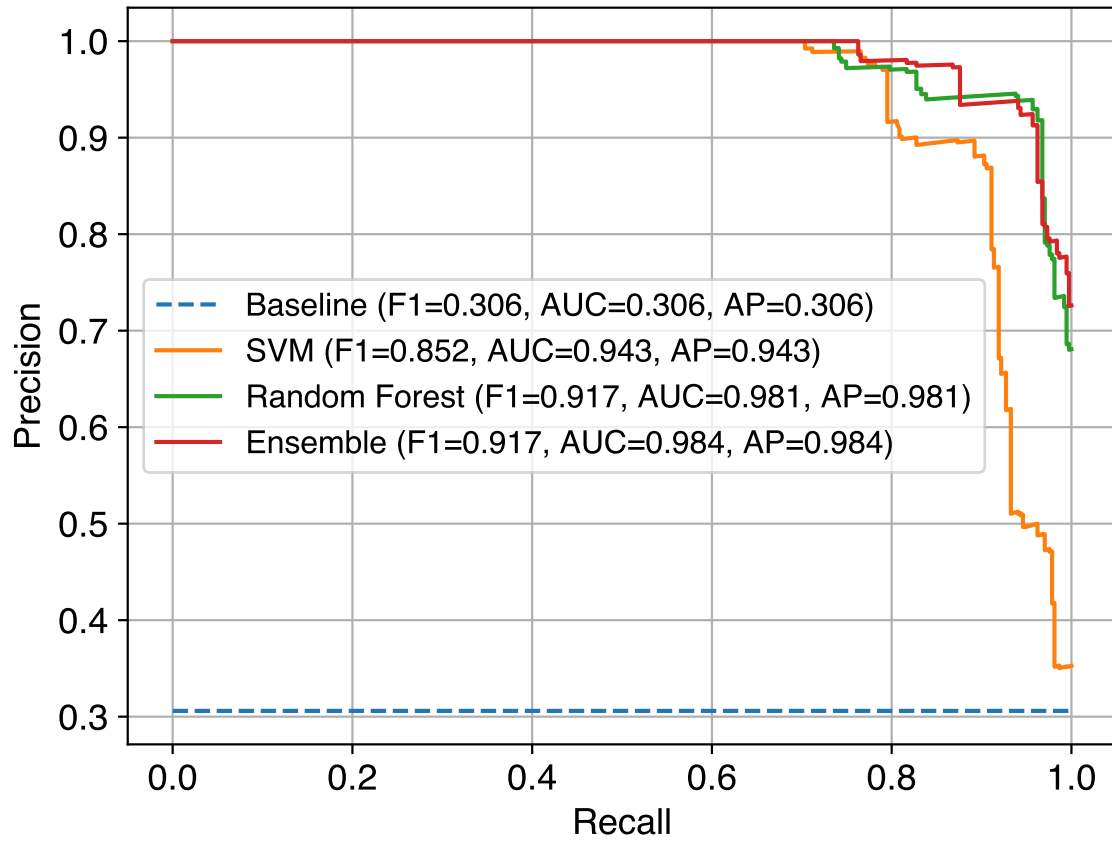


Fig. 6.4.: PR curves, F1 score, AUC score, and AP score on the test set for all the trained models using 10% of the available training data (68 videos).

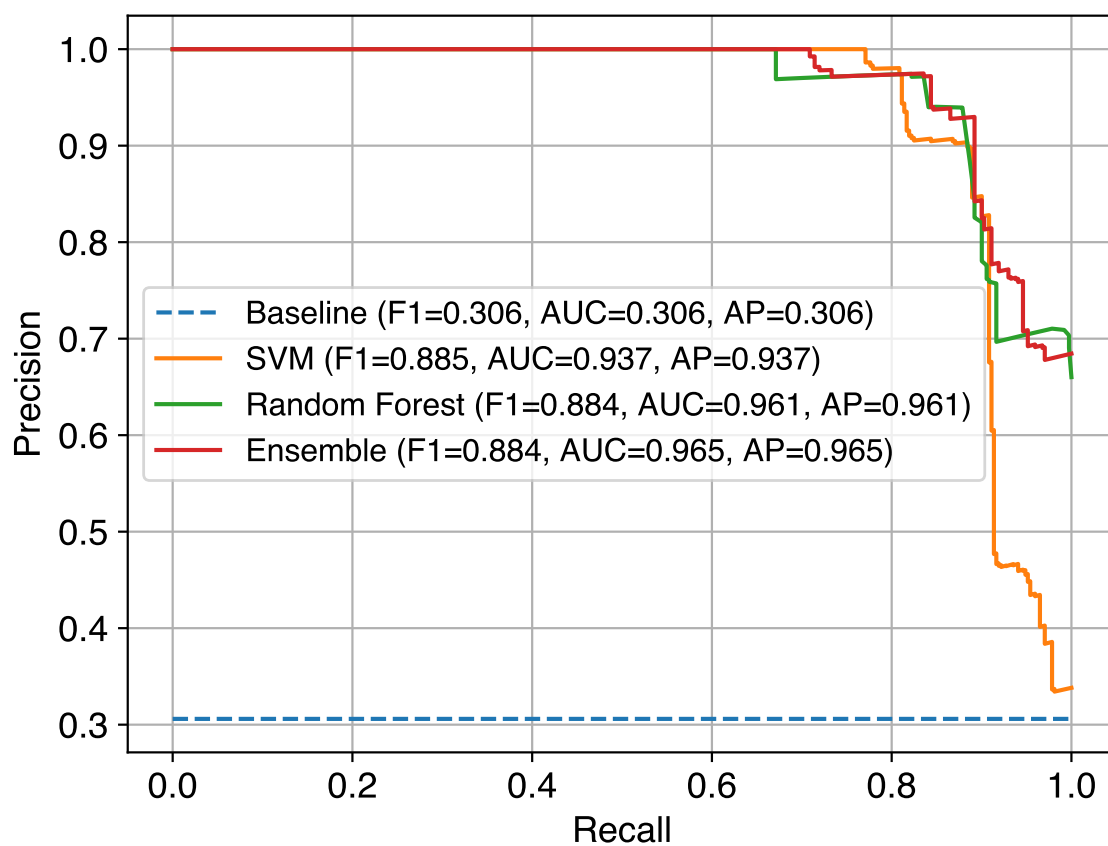


Fig. 6.5.: PR curves, F1 score, AUC score, and AP score on the test set for all the trained models using 25% of the available training data (169 videos).

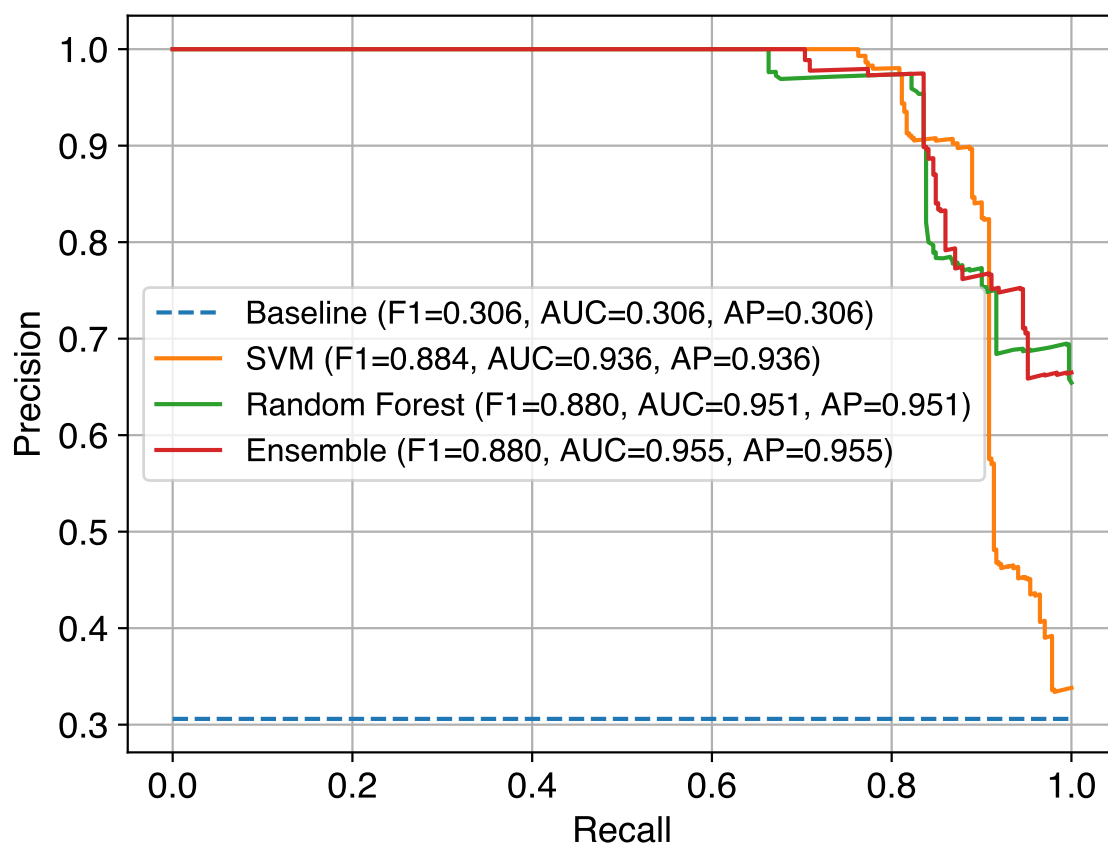


Fig. 6.6.: PR curves, F1 score, AUC score, and AP score on the test set for all the trained models using 50% of the available training data (339 videos).

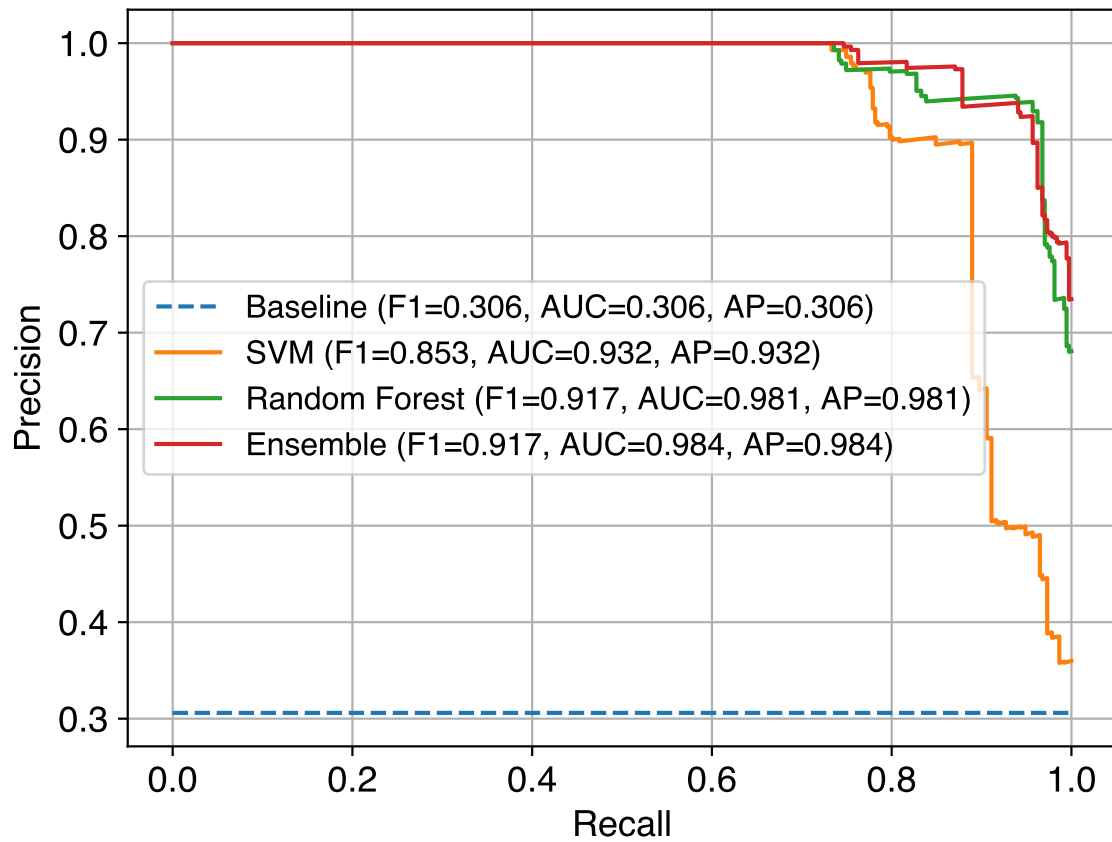


Fig. 6.7.: PR curves, F1 score, AUC score, and AP score on the test set for all the trained models using 75% of the available training data (508 videos).

## 7. SUMMARY AND FUTURE WORK

### 7.1 Overview

In Chapter 2 we showed the possibility of using CNNs for camera model identification. Specifically, we focused on a processing pipeline making use of a CNN for feature extraction and a set of SVMs for classification. We first tested different CNN architectures, in order to select a valid structure candidate balancing accuracy and computational complexity. We then investigated the effect of training a CNN on different data splits in order to highlight the dependency between accuracy, training set size, and training-testing splitting policy.

Our study shows that it is possible to achieve high camera model attribution accuracy (i.e., around 96% ) even with fairly small network architectures (i.e., four convolutional layers), provided that a minimum amount of training images are available. Indeed, the use of larger configurations determines a negligible accuracy increment, at least on the selected dataset of 18 camera models.

In Chapter 3 we presented a method for reliability patch estimation for camera model attribution. This means being able to estimate the likelihood that an image patch will be correctly attributed to the camera model used to acquire the image from which the patch comes from.

The proposed solution is based on concatenating a pre-trained CNN for patch-wise camera model attribution with a dense network that acts as a binary classifier. Exploiting transfer learning techniques and a two-tiered training strategy, it is possible to achieve 86% of accuracy in patch reliability estimation. Moreover, by running camera model attribution on single selected patches, camera attribution accuracy increases by more than 8%.

In addition to the impact on camera model attribution, the proposed method returns a reliability mask that highlights which image regions are considered reliable in terms of camera attribution. This could be useful in the future to better understand which image details are more important to camera model attribution CNNs. Moreover, it could be paired with splicing localization algorithms based on camera model traces to possibly opt-out unreliable regions from the analysis.

In Chapter 4 we described a counter-forensic method to subtly alter images to change their estimated camera model when they are analyzed by a CNN-based camera model detector. We tested our method on a reference dataset with images from multiple cameras that show highly similar indoor and outdoor scenes. The results demonstrate that we can generate imperceptibly altered adversarial images that are misclassified with high confidence by the CNN. In the future, we will extend our method to apply it to video sequences and we will explore viable adversarial example detection methods and defense techniques to increase the robustness of CNN-based camera model detectors.

In Chapter 5 we have presented a temporal-aware system to automatically detect deep-fake videos. Our experimental results using a large collection of manipulated videos have shown that using a simple convolutional LSTM structure we can accurately predict if a video has been subject to manipulation or not with as few as 2 seconds of video data.

We believe that our work offers a powerful first line of defense to spot fake media created using the tools described in the paper. We show how our system can achieve competitive results in this task while using a simple pipeline architecture. In future work, we plan to explore how to increase the robustness of our system against manipulated videos using unseen techniques during training.

Finally, in Chapter 6, we have shown how simple machine learning classifiers can be highly effective at detecting video manipulations when the appropriate data is used. Up until now, most video manipulation detection techniques have focused on analyzing the pixel data to spot forged content. More specifically, we use an ensemble of a random for-

est and an SVM trained on multimedia stream descriptors from both forged and pristine videos. With this approach, we have achieved an extremely high video manipulation detection score while requiring very limited amounts of data. Based on our findings, our future work will focus on techniques that automatically perform data sanitization. This will allow us to remove metadata and auxiliary header information that may give away sensitive information such as the source of the video.

## 7.2 Complete List Of Publications

Following is an exhaustive list of the publications in which I have been personally involved during my Ph.D. studies at Purdue University as a member of the VIPER Laboratory:

1. L. Bondi, L. Baroffio, **D. Güera**, P. Bestagini, E. J. Delp, S. Tubaro, “First Steps Toward Camera Model Identification With Convolutional Neural Networks”, *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 259-263, December 2016.  
URL: <https://doi.org/10.1109/LSP.2016.2641006>
2. L. Bondi, **D. Güera**, L. Baroffio, P. Bestagini, E. J. Delp, S. Tubaro, “A Preliminary Study on Convolutional Neural Networks for Camera Model Identification”, *Proceedings of the IS&T Electronic Imaging*, vol. 2017, no. 7, pp. 67-76, Burlingame, CA, January 2017.  
URL: <https://doi.org/10.2352/ISSN.2470-1173.2017.7.MWWSF-327>
3. L. Bondi, S. Lameri, **D. Güera**, P. Bestagini, E. J. Delp, S. Tubaro, “Tampering Detection And Localization Through Clustering Of Camera-Based CNN Features”, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1855-1864, Honolulu, HI, July 2017.  
URL: <https://doi.org/10.1109/CVPRW.2017.232>

4. **D. Güera**, Y. Wang, L. Bondi, P. Bestagini, S. Tubaro, E. J. Delp, “A Counter-Forensic Method For CNN-Based Camera Model Identification”, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1840-1847, Honolulu, HI, July 2017.  
URL: <https://doi.org/10.1109/CVPRW.2017.230>
5. K. Tahboub, **D. Güera**, A. R. Reibman, E. J. Delp, “Quality-Adaptive Deep Learning For Pedestrian Detection”, *Proceedings of the IEEE International Conference on Image Processing*, pp. 4187-4191, Beijing, China, September 2017.  
URL: <http://doi.org/10.1109/ICIP.2017.8297071>
6. S. K. Yarlagadda, **D. Güera**, P. Bestagini, F. Zhu, S. Tubaro, E. J. Delp, “Satellite Image Forgery Detection and Localization Using GAN and One-Class Classifier”, *Proceedings of the IS&T Electronic Imaging*, vol. 2018, no. 7, pp. 214-1-214-9, Burlingame, CA, January 2018.  
URL: <https://doi.org/10.2352/ISSN.2470-1173.2018.07.MWSF-214>
7. **D. Güera**, S. K. Yarlagadda, P. Bestagini, F. Zhu, S. Tubaro, E. J. Delp, “Reliability Map Estimation For CNN-Based Camera Model Attribution”, *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, pp. 964-973, Lake Tahoe, NV, February 2018.  
URL: <https://doi.org/10.1109/WACV.2018.00111>
8. N. Bonettini, L. Bondi, **D. Güera**, S. Mandelli, P. Bestagini, S. Tubaro, E. J. Delp, “Fooling PRNU-Based Detectors Through Convolutional Neural Networks”, *Proceedings of the IEEE European Signal Processing Conference*, pp. 957-961, Rome, Italy, September 2018.  
URL: <https://doi.org/10.23919/EUSIPCO.2018.8553596>
9. **D. Güera**, E. J. Delp, “Deepfake Video Detection Using Recurrent Neural Networks”, *Proceedings of the IEEE International Conference on Advanced Video and*



*Signal-based Surveillance*, pp. 1-6, Auckland, New Zealand, November 2018.

URL: <https://doi.org/10.1109/AVSS.2018.8639163>

10. E. R. Bartusiak, S. K. Yarlagadda, **D. Güera**, F. Zhu, P. Bestagini, S. Tubaro, E. J. Delp, “Splicing Detection And Localization In Satellite Imagery Using Conditional GANs”, *Proceedings of the IEEE International Conference on Multimedia Information Processing and Retrieval*, pp. 91-96, San Jose, CA, March 2019.  
URL: <https://doi.org/10.1109/MIPR.2019.00024>
11. S. K. Yarlagadda, **D. Güera**, D. Mas Montserrat, F. Zhu, P. Bestagini, S. Tubaro, E. J. Delp, “Shadow Removal Detection And Localization For Forensics Analysis”, *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 2677-2681, Brighton, UK, May 2019.  
URL: <https://doi.org/10.1109/ICASSP.2019.8683695>
12. H. Hao, **D. Güera**, J. Horváth, A. R. Reibman, E. J. Delp, “Robustness Analysis of Face Obscuration”, *arXiv:1905.05243v2*, June 2019.  
URL: <https://arxiv.org/abs/1905.05243v2>
13. H. Hao, **D. Güera**, A. R. Reibman, E. J. Delp, “A Utility-preserving GAN for Face Obscuration”, *Proceedings of the International Conference on Machine Learning, Synthetic Realities: Deep Learning for Detecting AudioVisual Fakes Workshop*, Long Beach, CA, June 2019.  
URL: <https://arxiv.org/abs/1906.11979>
14. **D. Güera**, S. Baireddy, P. Bestagini, S. Tubaro, E. J. Delp, “We Need No Pixels: Video Manipulation Detection Using Stream Descriptors”, *Proceedings of the International Conference on Machine Learning, Synthetic Realities: Deep Learning for Detecting AudioVisual Fakes Workshop*, Long Beach, CA, June 2019.  
URL: <https://arxiv.org/abs/1906.08743>
15. J. Horváth, **D. Güera**, S. K. Yarlagadda, P. Bestagini, F. Zhu, S. Tubaro, E. J. Delp, “Anomaly-based Manipulation Detection in Satellite Images”, *Proceedings of the*

*IEEE Conference on Computer Vision and Pattern Recognition, Workshop on Media Forensics*, pp. 62-71, Long Beach, CA, June 2019.

URL: [http://openaccess.thecvf.com/content\\_CVPRW\\_2019/html/Media\\_Forensics/Horvath\\_Anomaly-Based\\_Manipulation\\_Detection\\_in\\_Satellite\\_Images\\_CVPRW\\_2019\\_paper.html](http://openaccess.thecvf.com/content_CVPRW_2019/html/Media_Forensics/Horvath_Anomaly-Based_Manipulation_Detection_in_Satellite_Images_CVPRW_2019_paper.html)

16. J. Prat, **D. Güera**, Y. Chen, E. J. Delp, “Locating Objects Without Bounding Boxes”, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6479-6489, Long Beach, CA, June 2019.

URL: [http://openaccess.thecvf.com/content\\_CVPR\\_2019/html/Ribera\\_Locating\\_Objects\\_Without\\_Bounding\\_Boxes\\_CVPR\\_2019\\_paper.html](http://openaccess.thecvf.com/content_CVPR_2019/html/Ribera_Locating_Objects_Without_Bounding_Boxes_CVPR_2019_paper.html)

17. N. Bonettini, **D. Güera**, L. Bondi, P. Bestagini, E. J. Delp, S. Tubaro, “Image Anonymization Detection With Deep Handcrafted Features”, *Proceedings of the IEEE International Conference on Image Processing*, pp. 2304-2308, Taipei, Taiwan, September 2019.

URL: <https://doi.org/10.1109/ICIP.2019.8804294>

## REFERENCES

- [1] B. B. Zhu, M. D. Swanson, and A. H. Tewfik, "When seeing isn't believing [multimedia authentication technologies]," *IEEE Signal Processing Magazine*, vol. 21, no. 2, pp. 40–49, Mar. 2004. [Online]. Available: <https://doi.org/10.1109/MSP.2004.1276112>
- [2] H. Farid, "Digital doctoring: how to tell the real from the fake," *Significance*, vol. 3, no. 4, pp. 162–166, Nov. 2006. [Online]. Available: <https://doi.org/10.1111/j.1740-9713.2006.00197.x>
- [3] D. O'Sullivan, "When seeing is no longer believing," CNN Business <https://edition.cnn.com/interactive/2019/01/business/pentagons-race-against-deepfakes/>, (Accessed on 09/01/2019).
- [4] Y. Cao, L. Jia, Y. Chen, N. Lin, C. Yang, B. Zhang, Z. Liu, X. Li, and H. Dai, "Recent advances of generative adversarial networks in computer vision," *IEEE Access*, vol. 7, pp. 14 985–15 006, Dec. 2019. [Online]. Available: <https://doi.org/10.1109/ACCESS.2018.2886814>
- [5] A. Rocha, W. Scheirer, T. Boult, and S. Goldenstein, "Vision of the unseen: Current trends and challenges in digital image and video forensics," *ACM Computing Surveys*, vol. 43, no. 4, pp. 26:1–26:42, Oct. 2011. [Online]. Available: <https://doi.org/10.1145/1978802.1978805>
- [6] L. Bondi, L. Baroffio, D. Güera, P. Bestagini, E. J. Delp, and S. Tubaro, "First steps towards camera model identification with convolutional neural networks," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 259–263, Mar. 2017. [Online]. Available: <https://doi.org/10.1109/LSP.2016.2641006>
- [7] L. Bondi, D. Güera, L. Baroffio, P. Bestagini, E. J. Delp, and S. Tubaro, "A preliminary study on convolutional neural networks for camera model identification," *Proceedings of the IS&T International Symposium on Electronic Imaging*, Jan. 2017, Burlingame, CA. [Online]. Available: <https://doi.org/10.2352/ISSN.2470-1173.2017.7.MWSF-327>
- [8] D. Güera, S. K. Yarlagadda, P. Bestagini, F. Zhu, S. Tubaro, and E. J. Delp, "Reliability map estimation for cnn-based camera model attribution," *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, pp. 964–973, Mar. 2018, Lake Tahoe, NV. [Online]. Available: <https://doi.org/10.1109/WACV.2018.00111>
- [9] Deepttrace Labs - 2019 Report, "The state of deepfakes: Landscape, threats, and impact," Oct. 2019, (Accessed on 10/18/2019). [Online]. Available: <https://deeptancelabs.com/mapping-the-deepfake-landscape/>

- [10] D. Güera and E. J. Delp, “Deepfake video detection using recurrent neural networks,” *Proceedings of the IEEE International Conference on Advanced Video and Signal Based Surveillance*, pp. 1–6, Nov. 2018, Auckland, New Zealand. [Online]. Available: <https://doi.org/10.1109/AVSS.2018.8639163>
- [11] D. Güera, Y. Wang, L. Bondi, P. Bestagini, S. Tubaro, and E. J. Delp, “A counter-forensic method for CNN-based camera model identification,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1840–1847, Jul. 2017, Honolulu, HI. [Online]. Available: <https://doi.org/10.1109/CVPRW.2017.230>
- [12] D. Güera, S. Baireddy, P. Bestagini, S. Tubaro, and E. J. Delp, “We need no pixels: Video manipulation detection using stream descriptors,” *Proceedings of the International Conference on Machine Learning, Synthetic Realities: Deep Learning for Detecting AudioVisual Fakes Workshop*, Jun. 2019. [Online]. Available: <https://arxiv.org/abs/1906.08743>
- [13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” pp. 1097–1105, Dec. 2012, Lake Tahoe, NV. [Online]. Available: <https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks>
- [14] NIST, “Media forensics challenge,” Information Technology Laboratory / Information Access Division / Multimodal Information Group <https://www.nist.gov/itl/iad/mig/media-forensics-challenge>, (Accessed on 09/01/2019).
- [15] IEEE Signal Processing Society, “2018 IEEE signal processing cup: Forensic camera model identification challenge,” SP Cup <https://signalprocessingsociety.org/get-involved/signal-processing-cup>, (Accessed on 09/01/2019).
- [16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016. [Online]. Available: <https://www.deeplearningbook.org/>
- [17] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” *Neural computation*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006. [Online]. Available: <https://doi.org/10.1162/neco.2006.18.7.1527>
- [18] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, “Greedy layer-wise training of deep networks,” *Advances in Neural Information Processing Systems*, pp. 153–160, Dec. 2006, Vancouver, Canada. [Online]. Available: <https://dl.acm.org/citation.cfm?id=2976476>
- [19] M. Ranzato, C. Poultney, S. Chopra, and Y. LeCun, “Efficient learning of sparse representations with an energy-based model,” *Advances in Neural Information Processing Systems*, pp. 1137–1144, Dec. 2006, Vancouver, Canada. [Online]. Available: <https://dl.acm.org/citation.cfm?id=2976599>
- [20] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015. [Online]. Available: <https://doi.org/10.1038/nature14539>
- [21] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *Proceedings of the International Conference on Learning Representations*, May 2015, San Diego, CA. [Online]. Available: <https://arxiv.org/abs/1409.1556>

- [22] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, Jun. 2015, Boston, MA. [Online]. Available: <https://doi.org/10.1109/CVPR.2015.7298594>
- [23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, Jun. 2016, Las Vegas, NV. [Online]. Available: <https://doi.org/10.1109/CVPR.2016.90>
- [24] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Aug. 2016, Honolulu, HI. [Online]. Available: <https://doi.org/10.1109/CVPR.2017.243>
- [25] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1800–1807, Jul. 2017, Honolulu, HI. [Online]. Available: <https://doi.org/10.1109/CVPR.2017.195>
- [26] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7132–7141, Jun. 2018, Salt Lake City, UT. [Online]. Available: <https://doi.org/10.1109/CVPR.2018.00745>
- [27] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 539–546, Jun. 2005, San Diego, CA. [Online]. Available: <https://doi.org/10.1109/CVPR.2005.202>
- [28] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *arXiv:1901.06032v2*, Apr. 2019. [Online]. Available: <https://arxiv.org/abs/1901.06032>
- [29] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, and T. Chen, "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, no. C, pp. 354–377, May 2018. [Online]. Available: <https://doi.org/10.1016/j.patcog.2017.10.013>
- [30] S. Suwajanakorn, S. M. Seitz, and I. Kemelmacher-Shlizerman, "Synthesizing obama: Learning lip sync from audio," *ACM Transactions on Graphics*, vol. 36, no. 4, pp. 95:1–95:13, Jul. 2017. [Online]. Available: <https://doi.org/10.1145/3072959.3073640>
- [31] H. Kim, P. Garrido, A. Tewari, W. Xu, J. Thies, M. Niessner, P. Pérez, C. Richardt, M. Zollhöfer, and C. Theobalt, "Deep video portraits," *ACM Transactions on Graphics*, vol. 37, no. 4, pp. 163:1–163:14, Jul. 2018. [Online]. Available: <https://doi.org/10.1145/3197517.3201283>
- [32] J. Thies, M. Zollhöfer, C. Theobalt, M. Stamminger, and M. Niessner, "Headon: Real-time reenactment of human portrait videos," *ACM Transactions on Graphics*, vol. 37, no. 4, pp. 164:1–164:13, Jul. 2018. [Online]. Available: <http://doi.org/10.1145/3197517.3201350>

- [33] A. Pumarola, A. Agudo, A. M. Martinez, A. Sanfeliu, and F. Moreno-Noguer, "GANimation: Anatomically-aware facial animation from a single image," *Proceedings of the European Conference on Computer Vision*, pp. 818–833, Oct. 2018, Munich, Germany. [Online]. Available: [https://doi.org/10.1007/978-3-030-01249-6\\_50](https://doi.org/10.1007/978-3-030-01249-6_50)
- [34] K. Nagano, J. Seo, J. Xing, L. Wei, Z. Li, S. Saito, A. Agarwal, J. Fursund, and H. Li, "pagan: Real-time avatars using dynamic textures," *ACM Transactions on Graphics*, vol. 37, no. 6, pp. 258:1–258:12, Dec. 2018. [Online]. Available: <http://doi.org/10.1145/3272127.3275075>
- [35] C. Chan, S. Ginosar, T. Zhou, and A. A. Efros, "Everybody dance now," *Proceedings of the IEEE International Conference on Computer Vision*, Oct. 2019, Seoul, South Korea. [Online]. Available: <https://arxiv.org/abs/1808.07371v2>
- [36] A. Jones, J.-Y. Chiang, A. Ghosh, M. L. M. Hullin, J. Busch, and P. Debevec, "Real-time geometry and reflectance capture for digital face replacement," *White Paper*, 2008, ICT Technical Report ICT-TR-04-2008. University of Southern California Centers for Creative Technologies .
- [37] K. Dale, K. Sunkavalli, M. K. Johnson, D. Vlasic, W. Matusik, and H. Pfister, "Video face replacement," *ACM Transactions on Graphics*, vol. 30, no. 6, pp. 1–130, Dec. 2011. [Online]. Available: <https://doi.org/10.1145/2070781.2024164>
- [38] P. Garrido, L. Valgaerts, O. Rehmsen, T. Thormaehlen, P. Perez, and C. Theobalt, "Automatic face reenactment," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4217–4224, June 2014, Columbus, OH.
- [39] J. Thies, M. Zollhöfer, M. Stamminger, C. Theobalt, and M. Nießner, "Face2Face: Real-time face capture and reenactment of RGB videos," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2387–2395, Jun. 2016, Las Vegas, NV. [Online]. Available: <https://doi.org/10.1109/CVPR.2016.262>
- [40] M. Zollhöfer, J. Thies, P. Garrido, D. Bradley, T. Beeler, P. Pérez, M. Stamminger, M. Nießner, and C. Theobalt, "State of the art on monocular 3D face reconstruction, tracking, and applications," *Computer Graphics*, vol. 37, no. 2, pp. 523–550, May 2018. [Online]. Available: <https://doi.org/10.1111/cgf.13382>
- [41] W. Wang and H. Farid, "Exposing digital forgeries in interlaced and deinterlaced video," *IEEE Transactions on Information Forensics and Security*, vol. 2, no. 3, Sep. 2007. [Online]. Available: <https://doi.org/10.1109/TIFS.2007.902661>
- [42] P. Mullan, D. Cozzolino, L. Verdoliva, and C. Riess, "Residual-based forensic comparison of video sequences," *Proceedings of the IEEE International Conference on Image Processing*, pp. 1507–1511, Sep. 2017, Beijing, China. [Online]. Available: <https://doi.org/10.1109/ICIP.2017.8296533>
- [43] X. Ding, G. Yang, R. Li, L. Zhang, Y. Li, and X. Sun, "Identification of motion-compensated frame rate up-conversion based on residual signals," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 7, pp. 1497–1512, Jul. 2018. [Online]. Available: <https://doi.org/10.1109/TCSVT.2017.2676162>
- [44] P. Bestagini, S. Milani, M. Tagliasacchi, and S. Tubaro, "Local tampering detection in video sequences," *Proceedings of the IEEE International Workshop on Multimedia Signal Processing*, pp. 488–493, Sep. 2013, Pula, Italy. [Online]. Available: <https://doi.org/10.1109/MMSP.2013.6659337>

- [45] V. Conotter, E. Bodnari, G. Boato, and H. Farid, "Physiologically-based detection of computer generated faces in video," *Proceedings of the IEEE International Conference on Image Processing*, pp. 248–252, Oct. 2014, Paris, France. [Online]. Available: <https://doi.org/10.1109/ICIP.2014.7025049>
- [46] R. Raghavendra, K. B. Raja, S. Venkatesh, and C. Busch, "Transferable deep-cnn features for detecting digital and print-scanned morphed face images," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1822–1830, Jul. 2017, Honolulu, HI. [Online]. Available: <https://doi.org/10.1109/CVPRW.2017.228>
- [47] T. Carvalho, F. A. Faria, H. Pedrini, R. da S. Torres, and A. Rocha, "Illuminant-based transformed spaces for image forensics," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 4, pp. 720–733, Apr. 2016.
- [48] P. Zhou, X. Han, V. I. Morariu, and L. S. Davis, "Two-stream neural networks for tampered face detection," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1831–1839, Jul. 2017, Honolulu, HI. [Online]. Available: <https://doi.org/10.1109/CVPRW.2017.229>
- [49] Y. Li, M. Chang, and S. Lyu, "In ictu oculi: Exposing AI created fake videos by detecting eye blinking," *Proceedings of the IEEE International Workshop on Information Forensics and Security*, pp. 1–7, Dec. 2018, Hong Kong, China. [Online]. Available: <https://doi.org/10.1109/WIFS.2018.8630787>
- [50] D. Afchar, V. Nozick, J. Yamagishi, and I. Echizen, "Mesonet: a compact facial video forgery detection network," Dec. 2018.
- [51] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner, "FaceForensics++: Learning to detect manipulated facial images," *Proceedings of the IEEE International Conference on Computer Vision*, Oct. 2019, Seoul, South Korea. [Online]. Available: <https://arxiv.org/abs/1901.08971v3>
- [52] T. Gloe, A. Fischer, and M. Kirchner, "Forensic analysis of video file formats," *Digital Investigation*, vol. 11, pp. S68–S76, Apr. 2014. [Online]. Available: <https://doi.org/10.1016/j.diin.2014.03.009>
- [53] H. Farid, "Seeing is not believing," *IEEE Spectrum*, vol. 46, no. 8, pp. 44–51, Aug. 2009. [Online]. Available: <https://doi.org/10.1109/MSPEC.2009.5186556>
- [54] A. Piva, "An overview on image forensics," *ISRN Signal Processing*, vol. 2013, p. 22, Jan. 2013. [Online]. Available: <https://doi.org/10.1155/2013/496701>
- [55] M. C. Stamm, Min Wu, and K. J. R. Liu, "Information Forensics: An Overview of the First Decade," *IEEE Access*, vol. 1, pp. 167–200, May 2013. [Online]. Available: <https://doi.org/10.1109/ACCESS.2013.2260814>
- [56] M. Kharrazi, H. T. Sencar, and N. Memon, "Blind source camera identification," *Proceedings of the IEEE International Conference on Image Processing*, vol. 1, pp. 709–712, Oct. 2004, Singapore, Singapore. [Online]. Available: <https://doi.org/10.1109/ICIP.2004.1418853>
- [57] T. Filler, J. Fridrich, and M. Goljan, "Using sensor pattern noise for camera model identification," *Proceedings of the IEEE International Conference on Image Processing*, pp. 1296–1299, Oct. 2008, San Diego, CA. [Online]. Available: <https://doi.org/10.1109/ICIP.2008.4712000>

- [58] M. Kirchner and T. Gloe, "Forensic camera model identification," in *Handbook of Digital Forensics of Multimedia Data and Devices*, A. T. S. Ho and S. Li, Eds. Wiley-Blackwell, 2015, pp. 329–374, Hoboken, NJ. [Online]. Available: <https://doi.org/10.1109/ICIP.2004.1418853>
- [59] A. Swaminathan, M. Wu, and K. J. R. Liu, "Digital image forensics via intrinsic fingerprints," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 1, pp. 101–117, Mar. 2008. [Online]. Available: <https://doi.org/10.1109/TIFS.2007.916010>
- [60] K. Choi, E. Lam, and K. Wong, "Automatic source camera identification using the intrinsic lens radial distortion," *Optics Express*, vol. 14, no. 24, pp. 11 551–11 565, Nov. 2006. [Online]. Available: <https://doi.org/10.1364/OE.14.011551>
- [61] S. Bayram, H. Sencar, N. Memon, and I. Avcibas, "Source camera identification based on CFA interpolation," *Proceedings of the IEEE International Conference on Image Processing*, pp. III–69–72, Sep. 2005, Genova, Italy. [Online]. Available: <https://doi.org/10.1109/ICIP.2005.1530330>
- [62] H. Cao and A. C. Kot, "Accurate detection of demosaicing regularity for digital image forensics," *IEEE Transactions on Information Forensics and Security*, vol. 4, no. 4, pp. 899–910, Dec. 2009. [Online]. Available: <https://doi.org/10.1109/TIFS.2009.2033749>
- [63] S. Milani, P. Bestagini, M. Tagliasacchi, and S. Tubaro, "Demosaicing strategy identification via eigenalgorithms," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2014. [Online]. Available: <https://doi.org/10.1109/ICASSP.2014.6854082>
- [64] A. Dirik, H. Sencar, and N. Memon, "Digital single lens reflex camera identification from traces of sensor dust," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 3, pp. 539–552, Sep. 2008. [Online]. Available: <https://doi.org/10.1109/TIFS.2008.926987>
- [65] G. Xu and Y. Q. Shi, "Camera model identification using local binary patterns," *Proceedings of the IEEE International Conference on Multimedia and Expo*, pp. 392–397, Jul. 2012, Melbourne, Australia. [Online]. Available: <https://doi.org/10.1109/ICME.2012.87>
- [66] C. Chen and M. C. Stamm, "Camera model identification framework using an ensemble of demosaicing features," *Proceedings of the IEEE International Workshop on Information Forensics and Security*, pp. 1–6, Nov. 2015, Rome, Italy. [Online]. Available: <https://doi.org/10.1109/WIFS.2015.7368573>
- [67] F. Marra, G. Poggi, C. Sansone, and L. Verdoliva, "Evaluation of residual-based local features for camera model identification," in *New Trends in Image Analysis and Processing – ICIAP 2015 Workshops*, V. Murino, E. Puppo, D. Sona, M. Cristani, and C. Sansone, Eds. Springer International Publishing, 2015, pp. 11–18, Cham, Switzerland. [Online]. Available: [https://doi.org/10.1007/978-3-319-23222-5\\\_2](https://doi.org/10.1007/978-3-319-23222-5\_2)
- [68] A. Tuama, F. Comby, and M. Chaumont, "Source camera model identification using features from contaminated sensor noise," *Proceedings of the International Workshop on Digital-Forensics and Watermarking*, pp. 83–93, Oct. 2016, Tokyo, Japan. [Online]. Available: [https://doi.org/10.1007/978-3-319-31960-5\\\_8](https://doi.org/10.1007/978-3-319-31960-5\_8)



- [69] Y. Bengio, "Learning Deep Architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, Jan. 2009. [Online]. Available: <https://doi.org/10.1561/22000000006>
- [70] M. Buccoli, P. Bestagini, M. Zanoni, A. Sarti, and S. Tubaro, "Unsupervised feature learning for bootleg detection using deep learning architectures," *Proceedings of the IEEE International Workshop on Information Forensics and Security*, pp. 131–136, Dec. 2014, Atlanta, GA. [Online]. Available: <https://doi.org/10.1109/WIFS.2014.7084316>
- [71] C. Jiansheng, K. Xiangui, L. Ye, and Z. J. Wang, "Median filtering forensics based on convolutional neural networks," *IEEE Signal Processing Letters*, vol. 22, no. 11, pp. 1849–1853, Nov. 2015. [Online]. Available: <https://doi.org/10.1109/LSP.2015.2438008>
- [72] B. Bayar and M. C. Stamm, "A deep learning approach to universal image manipulation detection using a new convolutional layer," *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security*, pp. 5–10, Jun. 2016, Vigo, Spain. [Online]. Available: <https://doi.org/10.1145/2909827.2930786>
- [73] G. Xu, H. Z. Wu, and Y. Q. Shi, "Structural design of convolutional neural networks for steganalysis," *IEEE Signal Processing Letters*, vol. 23, no. 5, pp. 708–712, May 2016. [Online]. Available: <https://doi.org/10.1109/LSP.2016.2548421>
- [74] L. Baroffio, L. Bondi, P. Bestagini, and S. Tubaro, "Camera identification with deep convolutional networks," *arXiv:1603.01068*, Mar. 2016. [Online]. Available: <http://arxiv.org/abs/1603.01068>
- [75] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time-series," in *The Handbook of Brain Theory and Neural Networks*, M. A. Arbib, Ed. Cambridge, MA: MIT Press, 1995. [Online]. Available: <https://mitpress.mit.edu/books/handbook-brain-theory-and-neural-networks>
- [76] V. Nair and G. Hinton, "Rectified linear units improve restricted boltzmann machines," *Proceedings of the International Conference on Machine Learning*, pp. 807–814, Jun. 2010, Haifa, Israel. [Online]. Available: <https://dl.acm.org/citation.cfm?id=3104322.3104425>
- [77] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, Jan. 2014. [Online]. Available: <http://jmlr.org/papers/v15/srivastava14a.html>
- [78] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, Jun. 2016, Las Vegas, NV. [Online]. Available: <https://doi.org/10.1109/CVPR.2016.308>
- [79] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," *Proceedings of the British Machine Vision Conference*, Sep. 2014, Nottingham, UK. [Online]. Available: <https://doi.org/10.5244/C.28.6>

- [80] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, Dec. 2015. [Online]. Available: <https://doi.org/10.1007/s11263-015-0816-y>
- [81] T. Gloe and R. Böhme, "The Dresden image database for benchmarking digital image forensics," *Journal of Digital Forensic Practice*, vol. 3, no. 2-4, pp. 150–159, Dec. 2010. [Online]. Available: <https://doi.org/10.1080/15567281.2010.531500>
- [82] D. Cozzolino, D. Gragnaniello, and L. Verdoliva, "Image forgery localization through the fusion of camera-based, feature-based and pixel-based techniques," *Proceedings of the IEEE International Conference on Image Processing*, pp. 5302–5306, Oct. 2014, Paris, France. [Online]. Available: <https://doi.org/10.1109/ICIP.2014.7026073>
- [83] J. Lukáš, J. Fridrich, and M. Goljan, "Digital camera identification from sensor pattern noise," *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 2, pp. 205–214, Jun. 2006. [Online]. Available: <https://doi.org/10.1109/TIFS.2006.873602>
- [84] M. Goljan and J. Fridrich, "Camera identification from cropped and scaled images," *Proceedings of the SPIE Electronic Imaging*, vol. 6819, Jan. 2008, San Jose, CA. [Online]. Available: <https://doi.org/10.1117/12.766732>
- [85] G. Cao, Y. Zhao, R. Ni, L. Yu, and H. Tian, "Forensic detection of median filtering in digital images," *Proceedings of the IEEE International Conference on Multimedia and Expo*, pp. 89–94, Jul. 2010, Singapore. [Online]. Available: <https://doi.org/10.1109/ICME.2010.5583869>
- [86] X. Zhao and M. C. Stamm, "Computationally efficient demosaicing filter estimation for forensic camera model identification," *Proceedings of the IEEE International Conference on Image Processing*, pp. 151–155, Sep. 2016, Phoenix, AZ. [Online]. Available: <https://doi.org/10.1109/ICIP.2016.7532337>
- [87] S.-H. Chen and C.-T. Hsu, "Source camera identification based on camera gain histogram," *Proceedings of the IEEE International Conference on Image Processing*, pp. 429–432, Sep. 2007, San Antonio, TX. [Online]. Available: <https://doi.org/10.1109/ICIP.2007.4380046>
- [88] T. H. Thai, R. Cograñne, and F. Retraint, "Camera model identification based on the heteroscedastic noise model," *IEEE Transactions on Image Processing*, vol. 23, no. 1, pp. 250–263, Jan. 2014. [Online]. Available: <https://doi.org/10.1109/TIP.2013.2290596>
- [89] A. Tuama, F. Comby, and M. Chaumont, "Camera model identification based on machine learning approach with high order statistics features," *Proceedings of the IEEE European Signal Processing Conference*, pp. 1183–1187, Aug. 2016, Budapest, Hungary. [Online]. Available: <https://doi.org/10.1109/EUSIPCO.2016.7760435>
- [90] L. Bondi, S. Lameri, D. Güera, P. Bestagini, E. J. Delp, and S. Tubaro, "Tampering detection and localization through clustering of camera-based cnn features," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1855–1864, Jul. 2017, Honolulu, HI. [Online]. Available: <https://doi.org/10.1109/CVPRW.2017.232>

- [91] Y. Chen, F. He, Y. Wu, and N. Hou, "A local start search algorithm to compute exact hausdorff distance for arbitrary point sets," *Pattern Recognition*, vol. 67, no. Supplement C, pp. 139–148, Jul. 2017. [Online]. Available: <https://doi.org/10.1016/j.patcog.2017.02.013>
- [92] A. Tuama, F. Comby, and M. Chaumont, "Camera model identification with the use of deep convolutional neural networks," *Proceedings of the IEEE International Workshop on Information Forensics and Security*, pp. 1–6, Dec. 2016, Abu Dhabi, United Arab Emirates. [Online]. Available: <https://doi.org/10.1109/WIFS.2016.7823908>
- [93] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Proceedings of the International Conference on Learning Representations*, May 2015, San Diego, CA. [Online]. Available: <https://arxiv.org/abs/1412.6980>
- [94] L. N. Smith, "Cyclical learning rates for training neural networks," *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, pp. 464–472, Mar. 2017, Santa Rosa, CA. [Online]. Available: <https://doi.org/10.1109/WACV.2017.58>
- [95] K. Liang, J. K. Liu, R. Lu, and D. S. Wong, "Privacy concerns for photo sharing in online social networks," *IEEE Internet Computing*, vol. 19, no. 2, pp. 58–63, Mar. 2015. [Online]. Available: <https://doi.org/10.1109/MIC.2014.107>
- [96] P. J. Chiang, N. Khanna, A. K. Mikkilineni, M. V. O. Segovia, S. Suh, J. P. Allebach, G. T. C. Chiu, and E. J. Delp, "Printer and scanner forensics," *IEEE Signal Processing Magazine*, vol. 26, no. 2, pp. 72–83, Mar. 2009. [Online]. Available: <https://doi.org/10.1109/MSP.2008.931082>
- [97] N. Khanna, A. K. Mikkilineni, A. F. Martone, G. N. Ali, G. T.-C. Chiu, J. P. Allebach, and E. J. Delp, "A survey of forensic characterization methods for physical devices," *Digital Investigation*, vol. 3, pp. 17–28, Sep. 2006. [Online]. Available: <https://doi.org/10.1016/j.diin.2006.06.014>
- [98] N. Khanna, A. K. Mikkilineni, and E. J. Delp, "Scanner identification using feature-based processing and analysis," *IEEE Transactions on Information Forensics and Security*, vol. 4, no. 1, pp. 123–139, Mar. 2009. [Online]. Available: <https://doi.org/10.1109/TIFS.2008.2009604>
- [99] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1725–1732, Jun. 2014, Columbus, OH. [Online]. Available: <https://doi.org/10.1109/CVPR.2014.223>
- [100] D. Mas Montserrat, Q. Lin, J. Allebach, and E. J. Delp, "Training object detection and recognition CNN models using data augmentation," *Proceedings of the IS&T International Symposium on Electronic Imaging*, Jan. 2017, Burlingame, CA. [Online]. Available: <https://doi.org/10.2352/ISSN.2470-1173.2017.10.IMAWM-163>
- [101] M. Kirchner and R. Böhme, "Tamper hiding: Defeating image forensics," *Proceedings of the International Workshop on Information Hiding*, pp. 326–341, Jun. 2007, Saint-Malo, France. [Online]. Available: [https://doi.org/10.1007/978-3-540-77370-2\\_22](https://doi.org/10.1007/978-3-540-77370-2_22)

- [102] R. Böhme and M. Kirchner, “Counter-forensics: Attacking image forensics,” in *Digital Image Forensics: There is More to a Picture than Meets the Eye*, H. T. Sencar and N. Memon, Eds. New York, NY: Springer New York, 2013, pp. 327–366. [Online]. Available: [https://doi.org/10.1007/978-1-4614-0757-7\\_12](https://doi.org/10.1007/978-1-4614-0757-7_12)
- [103] M. Goljan, J. Fridrich, and M. Chen, “Sensor noise camera identification: Countering counter-forensics,” *Proceedings of the IS&T/SPIE Electronic Imaging Conference*, pp. 75 410S–75 410S, Jan. 2010, San Jose, CA. [Online]. Available: <https://doi.org/10.1117/12.839055>
- [104] T. Gloe, M. Kirchner, A. Winkler, and R. Böhme, “Can we trust digital image forensics?” *Proceedings of the ACM International Conference on Multimedia*, pp. 78–86, Sep. 2007, Augsburg, Germany. [Online]. Available: <https://doi.org/10.1145/1291233.1291252>
- [105] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” *Proceedings of the International Conference on Learning Representations*, Apr. 2014, Banff, Canada. [Online]. Available: <https://arxiv.org/abs/1312.6199>
- [106] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” *Proceedings of the International Conference on Learning Representations*, May 2015, San Diego, CA. [Online]. Available: <https://arxiv.org/abs/1412.6572>
- [107] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, “Distillation as a defense to adversarial perturbations against deep neural networks,” *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 582–597, May 2016, San Jose, CA. [Online]. Available: <https://doi.org/10.1109/SP.2016.41>
- [108] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, “The limitations of deep learning in adversarial settings,” *Proceedings of the IEEE European Symposium on Security and Privacy*, pp. 372–387, Mar. 2016, Saarbrücken, Germany. [Online]. Available: <https://doi.org/10.1109/EuroSP.2016.36>
- [109] C.-C. J. Kuo, “Understanding convolutional neural networks with a mathematical model,” *Journal of Visual Communication and Image Representation*, vol. 41, pp. 406–413, Nov. 2016. [Online]. Available: <https://doi.org/10.1016/j.jvcir.2016.11.003>
- [110] L. Bottou, “Large-scale machine learning with stochastic gradient descent,” *Proceedings of the International Conference on Computational Statistics*, pp. 177–186, Aug. 2010, Paris, France. [Online]. Available: [https://doi.org/10.1007/978-3-7908-2604-3\\_16](https://doi.org/10.1007/978-3-7908-2604-3_16)
- [111] N. Papernot, I. Goodfellow, R. Sheatsley, R. Feinman, and P. McDaniel, “cleverhans v1.0.0: an adversarial machine learning library,” *arXiv:1610.00768v3*, Oct. 2016. [Online]. Available: <https://arxiv.org/abs/1610.00768v3>
- [112] S. Lorant, *Lincoln; a picture story of his life*. Norton, 1969. [Online]. Available: <https://www.amazon.com/dp/0393074463>
- [113] “Fakeapp,” <https://www.fakeapp.org/>, (Accessed on 05/29/2018).

- [114] P. Isola, J. Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5967–5976, Jul. 2017, Honolulu, HI. [Online]. Available: <https://doi.org/10.1109/CVPR.2017.632>
- [115] J. Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2242–2251, Oct. 2017, Venice, Italy. [Online]. Available: <https://doi.org/10.1109/ICCV.2017.244>
- [116] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, “Tensorflow: A system for large-scale machine learning,” *Proceedings of the USENIX Conference on Operating Systems Design and Implementation*, vol. 16, pp. 265–283, Nov. 2016, Savannah, GA. [Online]. Available: <https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi>
- [117] F. Chollet *et al.*, “Keras,” <https://keras.io>, 2015.
- [118] A. Tewari, M. Zollhöfer, H. Kim, P. Garrido, F. Bernard, P. Pérez, and C. Theobalt, “MoFA: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction,” *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3735–3744, Oct. 2017, Venice, Italy. [Online]. Available: <https://doi.org/10.1109/ICCV.2017.401>
- [119] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” *Advances in Neural Information Processing Systems*, pp. 2672–2680, Dec. 2014, Montréal, Canada. [Online]. Available: <http://papers.nips.cc/paper/5423-generative-adversarial-nets>
- [120] G. Antipov, M. Baccouche, and J.-L. Dugelay, “Face aging with conditional generative adversarial networks,” *arXiv:1702.01983v2*, Feb. 2017. [Online]. Available: <https://arxiv.org/abs/1702.01983v2>
- [121] “Faceapp,” <https://www.faceapp.com/>, (Accessed on 05/29/2018).
- [122] “What are deepfakes & why the future of porn is terrifying,” <https://www.highsnobiety.com/p/what-are-deepfakes-ai-porn/>, (Accessed on 05/29/2018).
- [123] “The Outline: Experts fear face swapping tech could start an international show-down,” <https://theoutline.com/post/3179/>, (Accessed on 05/29/2018).
- [124] M. Brundage, S. Avin, J. Clark, H. Toner, P. Eckersley, B. Garfinkel, A. Dafoe, P. Scharre, T. Zeitzoff, B. Filar, H. Anderson, H. Roff, G. C. Allen, J. Steinhardt, C. Flynn, S. Ó. hÉigeartaigh, S. Beard, H. Belfield, S. Farquhar, C. Lyle, R. Crootof, O. Evans, M. Page, J. Bryson, R. Yampolskiy, and D. Amodei, “The malicious use of artificial intelligence: Forecasting, prevention, and mitigation,” *arXiv:1802.07228v1*, Feb. 2018. [Online]. Available: <https://arxiv.org/abs/1802.07228v1>
- [125] H. T. Sencar and N. Memon, Eds., *Digital Image Forensics*. New York, NY: Springer New York, 2013. [Online]. Available: <https://doi.org/10.1007/978-1-4614-0757-7>

- [126] H. Farid, *Photo Forensics*. Cambridge, MA: MIT Press Ltd, 2016. [Online]. Available: <https://mitpress.mit.edu/books/photo-forensics>
- [127] N. Rahmouni, V. Nozick, J. Yamagishi, and I. Echizen, “Distinguishing computer graphics from natural images using convolution neural networks,” *Proceedings of the IEEE Workshop on Information Forensics and Security*, pp. 1–6, Dec. 2017, Rennes, France. [Online]. Available: <https://doi.org/10.1109/WIFS.2017.8267647>
- [128] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner, “Faceforensics: A large-scale video dataset for forgery detection in human faces,” *arXiv:1803.09179*, Mar. 2018. [Online]. Available: <https://arxiv.org/abs/1803.09179>
- [129] C. Bregler, M. Covell, and M. Slaney, “Video rewrite: Driving visual speech with audio,” *Proceedings of the ACM Annual Conference on Computer Graphics And Interactive Techniques*, pp. 353–360, Aug. 1997, Los Angeles, CA. [Online]. Available: <https://doi.org/10.1145/258734.258880>
- [130] H. Averbuch-Elor, D. Cohen-Or, J. Kopf, and M. F. Cohen, “Bringing portraits to life,” *ACM Transactions on Graphics*, vol. 36, no. 6, pp. 196:1–196:13, Nov. 2017. [Online]. Available: <https://doi.org/10.1145/3130800.3130818>
- [131] Z. Lu, Z. Li, J. Cao, R. He, and Z. Sun, “Recent progress of face image synthesis,” *arXiv:1706.04717v1*, Jun. 2017. [Online]. Available: <https://arxiv.org/abs/1706.04717v1>
- [132] Y. Lu, Y.-W. Tai, and C.-K. Tang, “Attribute-guided face generation using conditional cycleGAN,” *Proceedings of the European Conference on Computer Vision*, pp. 293–308, Oct. 2018, Munich, Germany. [Online]. Available: [https://doi.org/10.1007/978-3-030-01258-8\\_18](https://doi.org/10.1007/978-3-030-01258-8_18)
- [133] P. Upchurch, J. Gardner, G. Pleiss, R. Pless, N. Snavely, K. Bala, and K. Weinberger, “Deep feature interpolation for image content changes,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6090–6099, Jul. 2017, Honolulu, HI. [Online]. Available: <https://doi.org/10.1109/CVPR.2017.645>
- [134] G. Lample, G. Lample, N. Zeghidour, N. Usunier, A. Bordes, L. Denoyer, and M. Ranzato, “Fader networks: Manipulating images by sliding attributes,” *Advances in Neural Information Processing Systems*, pp. 5967–5976, Dec. 2017, Long Beach, CA. [Online]. Available: <https://papers.nips.cc/paper/7178-fader-networksmanipulating-images-by-sliding-attributes>
- [135] T. Karras, T. Aila, S. Laine, and J. Lehtinen, “Progressive growing of GANs for improved quality, stability, and variation,” *Proceedings of the International Conference on Learning Representations*, Apr. 2018. [Online]. Available: <https://openreview.net/forum?id=Hk99zCeAb>
- [136] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: <https://doi.org/10.1162/neco.1997.9.8.1735>
- [137] J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell, “Long-term recurrent convolutional networks for visual recognition and description,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 677–691, Apr. 2017. [Online]. Available: <https://doi.org/10.1109/TPAMI.2016.2599174>

- [138] N. McLaughlin, J. M. d. Rincon, and P. Miller, “Recurrent convolutional network for video-based person re-identification,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1325–1334, June 2016, Las Vegas, NV. [Online]. Available: <https://doi.org/10.1109/CVPR.2016.148>
- [139] Y. Liao, Y. Wang, and Y. Liu, “Graph regularized auto-encoders for image representation,” *IEEE Transactions on Image Processing*, vol. 26, no. 6, pp. 2839–2852, June 2017. [Online]. Available: <https://doi.org/10.1109/TIP.2016.2605010>
- [140] Y. Qian, K. Chen, J. Nikkanen, J. Kämäräinen, and J. Matas, “Recurrent color constancy,” *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5459–5467, Oct. 2017, Venice, Italy. [Online]. Available: <https://doi.org/10.1109/ICCV.2017.582>
- [141] “Kaggle/IEEE’s Signal Processing Society: Camera Model Identification,” <https://www.kaggle.com/c/sp-society-camera-model-identification/discussion/49299>, (Accessed on 05/29/2018).
- [142] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, “Learning realistic human actions from movies,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, Jun. 2008, Anchorage, AK. [Online]. Available: <https://doi.org/10.1109/CVPR.2008.4587756>
- [143] N. Dufour and A. Gully, “Contributing data to deepfake detection research,” Sep. 2019, (Accessed on 09/05/2019). [Online]. Available: <https://ai.googleblog.com/2019/09/contributing-data-to-deepfake-detection.html>
- [144] M. Schroepfer, “Creating a data set and a challenge for deepfakes,” Sep. 2019, (Accessed on 09/05/2019). [Online]. Available: <https://ai.facebook.com/blog/deepfake-detection-challenge/>
- [145] Facebook, “Deepfake detection challenge,” Sep. 2019, (Accessed on 09/05/2019). [Online]. Available: <https://deepfakedetectionchallenge.ai>
- [146] A. Bansal, S. Ma, D. Ramanan, and Y. Sheikh, “Recycle-GAN: Unsupervised video retargeting,” *Proceedings of the European Conference on Computer Vision*, pp. 119–135, Sep. 2018, Munich, Germany. [Online]. Available: [https://doi.org/10.1007/978-3-030-01228-1\\_8](https://doi.org/10.1007/978-3-030-01228-1_8)
- [147] P. Korshunov and S. Marcel, “Deepfakes: a new threat to face recognition? assessment and detection,” *arXiv:1812.08685v1*, Mar. 2018. [Online]. Available: <https://arxiv.org/abs/1812.08685v1>
- [148] I. Korshunova, W. Shi, J. Dambre, and L. Theis, “Fast face-swap using convolutional neural networks,” *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3697–3705, Oct. 2017, Venice, Italy. [Online]. Available: <https://doi.org/10.1109/ICCV.2017.397>
- [149] J. Vincent, “US lawmakers say AI deepfakes ‘have the potential to disrupt every facet of our society’,” Sep. 2018, (Accessed on 04/17/2019). [Online]. Available: <https://www.theverge.com/2018/9/14/17859188/ai-deepfakes-national-security-threat-lawmakers-letter-intelligence-community>

- [150] R. Chesney and D. K. Citron, “Disinformation on steroids: The threat of deep fakes,” Oct. 2018, (Accessed on 04/17/2019). [Online]. Available: <https://www.cfr.org/report/deep-fake-disinformation-steroids>
- [151] M. Bird, “The video in which Greece’s finance minister gives Germany the finger has several bizarre new twists,” Mar. 2015, (Accessed on 04/17/2019). [Online]. Available: <https://www.businessinsider.com/yanis-varoufakis-middle-finger-controversy-real-fake-bohmermann-jauch-2015-3>
- [152] C. Curtis, “Deepfakes are being weaponized to silence women — but this woman is fighting back,” Oct. 2018, (Accessed on 04/17/2019). [Online]. Available: <https://thenextweb.com/code-word/2018/10/05/deepfakes-are-being-weaponized-to-silence-women-but-this-woman-is-fighting-back/>
- [153] S. Cole, “Fake porn makers are worried about accidentally making child porn,” Feb. 2018, (Accessed on 04/17/2019). [Online]. Available: [https://motherboard.vice.com/en\\_us/article/evmkxa/ai-fake-porn-deepfakes-child-pornography-emma-watson-elle-fanning](https://motherboard.vice.com/en_us/article/evmkxa/ai-fake-porn-deepfakes-child-pornography-emma-watson-elle-fanning)
- [154] M. C. Stamm, W. S. Lin, and K. J. R. Liu, “Forensics vs. anti-forensics: A decision and game theoretic framework,” *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1749–1752, Mar. 2012, Kyoto, Japan. [Online]. Available: <https://doi.org/10.1109/ICASSP.2012.6288237>
- [155] K. Jack, “Chapter 13 - MPEG-2,” in *Video Demystified: A Handbook for the Digital Engineer*, K. Jack, Ed. Burlington, MA: Newnes, 2007, pp. 577–737. [Online]. Available: <https://doi.org/10.1016/B978-075068395-1/50013-4>
- [156] H. Guan, M. Kozak, E. Robertson, Y. Lee, A. N. Yates, A. Delgado, D. Zhou, T. Kheyrkhah, J. Smith, and J. Fiscus, “Mfc datasets: Large-scale benchmark datasets for media forensic challenge evaluation,” *Proceedings of the IEEE Winter Applications of Computer Vision Workshops*, pp. 63–72, Jan. 2019, Waikoloa Village, HI. [Online]. Available: <https://doi.org/10.1109/WACVW.2019.00018>
- [157] S. Milani, M. Fontani, P. Bestagini, M. Barni, A. Piva, M. Tagliasacchi, and S. Tubaro, “An overview on video forensics,” *APSIPA Transactions on Signal and Information Processing*, vol. 1, p. e2, Aug. 2012. [Online]. Available: <https://doi.org/10.1017/ATSIP.2012.2>
- [158] M. C. Stamm, W. S. Lin, and K. J. R. Liu, “Temporal forensics and anti-forensics for motion compensated video,” *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 4, pp. 1315–1329, Aug. 2012. [Online]. Available: <https://doi.org/10.1109/TIFS.2012.2205568>
- [159] S. Bian, W. Luo, and J. Huang, “Exposing fake bit rate videos and estimating original bit rates,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 12, pp. 2144–2154, Dec. 2014. [Online]. Available: <https://doi.org/10.1109/TCSVT.2014.2334031>
- [160] P. Bestagini, S. Milani, M. Tagliasacchi, and S. Tubaro, “Codec and gop identification in double compressed videos,” *IEEE Transactions on Image Processing*, vol. 25, no. 5, pp. 2298–2310, May 2016. [Online]. Available: <https://doi.org/10.1109/TIP.2016.2541960>



- [161] C.-C. Hsu, T.-Y. Hung, C.-W. Lin, and C.-T. Hsu, "Video forgery detection using correlation of noise residue," *Proceedings of IEEE Workshop on Multimedia Signal Processing*, pp. 170–174, Oct. 2008, Cairns, Qld, Australia. [Online]. Available: <https://doi.org/10.1109/MMSP.2008.4665069>
- [162] S. Mandelli, P. Bestagini, S. Tubaro, D. Cozzolino, and L. Verdoliva, "Blind detection and localization of video temporal splicing exploiting sensor-based footprints," *Proceedings of the European Signal Processing Conference*, pp. 1362–1366, Sep. 2018, Rome, Italy. [Online]. Available: <https://doi.org/10.23919/EUSIPCO.2018.8553511>
- [163] S. Bayram, H. T. Sencar, and N. Memon, "Video copy detection based on source device characteristics: A complementary approach to content-based methods," *Proceedings of the ACM International Conference on Multimedia Information Retrieval*, pp. 435–442, Oct. 2008, Vancouver, British Columbia, Canada. [Online]. Available: <https://doi.org/10.1145/1460096.1460167>
- [164] S. Lameri, L. Bondi, P. Bestagini, and S. Tubaro, "Near-duplicate video detection exploiting noise residual traces," *Proceedings of the IEEE International Conference on Image Processing*, pp. 1497–1501, Sep. 2017, Beijing, China. [Online]. Available: <https://doi.org/10.1109/ICIP.2017.8296531>
- [165] L. D'Amiano, D. Cozzolino, G. Poggi, and L. Verdoliva, "Video forgery detection and localization based on 3d PatchMatch," *Proceedings of the IEEE International Conference on Multimedia Expo Workshops*, pp. 1–6, Jun. 2015, Turin, Italy. [Online]. Available: <https://doi.org/10.1109/ICMEW.2015.7169805>
- [166] —, "A PatchMatch-based dense-field algorithm for video copy-move detection and localization," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 3, pp. 669–682, Mar. 2019. [Online]. Available: <https://doi.org/10.1109/TCSVT.2018.2804768>
- [167] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "Patchmatch: A randomized correspondence algorithm for structural image editing," *ACM Transactions on Graphics*, vol. 28, no. 3, pp. 24:1–24:11, Jul. 2009. [Online]. Available: <https://doi.org/10.1145/1531326.1531330>
- [168] D. D'Avino, D. Cozzolino, G. Poggi, and L. Verdoliva, "Autoencoder with recurrent neural networks for video forgery detection," *Proceedings of the IS&T Electronic Imaging*, vol. 2017, no. 7, pp. 92–99, Jan. 2017, Burlingame, CA. [Online]. Available: <https://doi.org/10.2352/ISSN.2470-1173.2017.7.MWSF-330>
- [169] F. Matern, C. Riess, and M. Stamminger, "Exploiting visual artifacts to expose deepfakes and face manipulations," *Proceedings of the IEEE Winter Applications of Computer Vision Workshops*, pp. 83–92, Jan. 2019, Waikoloa Village, HI. [Online]. Available: <https://doi.org/10.1109/WACVW.2019.00020>
- [170] N. Khanna, G. T. Chiu, J. P. Allebach, and E. J. Delp, "Forensic techniques for classifying scanner, computer generated and digital camera images," pp. 1653–1656, Mar. 2008, Las Vegas, NV. [Online]. Available: <https://doi.org/10.1109/ICASSP.2008.4517944>

- [171] T. Bianchi and A. Piva, "Image forgery localization via block-grained analysis of jpeg artifacts," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 1003–1017, Jun. 2012. [Online]. Available: <https://doi.org/10.1109/TIFS.2012.2187516>
- [172] O. Bulan, J. Mao, and G. Sharma, "Geometric distortion signatures for printer identification," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1401–1404, Apr. 2009. [Online]. Available: <https://doi.org/10.1109/ICASSP.2009.4959855>
- [173] J. Fan, A. C. Kot, H. Cao, and F. Sattar, "Modeling the exif-image correlation for image manipulation detection," *Proceedings of the IEEE International Conference on Image Processing*, pp. 1945–1948, Sep. 2011, Brussels, Belgium. [Online]. Available: <https://doi.org/10.1109/ICIP.2011.6115853>
- [174] M. Huh, A. Liu, A. Owens, and A. A. Efros, "Fighting fake news: Image splice detection via learned self-consistency," *Proceedings of the European Conference on Computer Vision*, pp. 106–124, Sep. 2018, Munich, Germany. [Online]. Available: [https://doi.org/10.1007/978-3-030-01252-6\\_7](https://doi.org/10.1007/978-3-030-01252-6_7)
- [175] M. Iuliani, D. Shullani, M. Fontani, S. Meucci, and A. Piva, "A video forensic framework for the unsupervised analysis of MP4-like file container," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 3, pp. 635–645, Mar. 2019. [Online]. Available: <https://doi.org/10.1109/TIFS.2018.2859760>
- [176] S. Zannettou, S. Chatzis, K. Papadamou, and M. Sirivianos, "The good, the bad and the bait: Detecting and characterizing clickbait on youtube," *Proceedings of the IEEE Security and Privacy Workshops*, pp. 63–69, May 2018, San Francisco, CA. [Online]. Available: <https://doi.org/10.1109/SPW.2018.00018>
- [177] F. Bellard *et al.*, "ffprobe documentation," Apr. 2019, (Accessed on 04/17/2019). [Online]. Available: <https://www.ffmpeg.org/ffprobe.html>
- [178] W. McKinney, "Data structures for statistical computing in python," *Proceedings of the Python in Science Conference*, pp. 51–56, Jun. 2010, Austin, TX. [Online]. Available: <http://conference.scipy.org/proceedings/scipy2010/mckinney.html>
- [179] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, Nov. 2011. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1953048.2078195>
- [180] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets," *PLoS ONE*, vol. 10, no. 3, p. e0118432, Mar. 2015. [Online]. Available: <https://doi.org/10.1371/journal.pone.0118432>

## VITA

David Güera was born in Barcelona, Catalonia, Spain in April 26, 1994. In July 2016, he received his Bachelor of Science in Telecommunications Systems Engineering from the Polytechnic University of Catalonia, at the Barcelona School Of Telecommunications Engineering. He was a visiting scholar in the Video and Image Processing Laboratory (VIPER) in Spring 2016. He also visited the Image and Sound Processing Group (ISPG) at the Politecnico di Milano in Spring 2019. While pursuing his Ph.D. at Purdue, he primarily worked on projects sponsored by the Air Force Research Laboratory (AFRL), the Defense Advanced Research Projects Agency (DARPA), and the National Geospatial Intelligence Agency (NGA). During his studies, Mr. Güera also gained industry experience with NVIDIA and Google.

His current research interests include machine and deep learning, image processing, and computer vision. He is a member of the IEEE, the IEEE Computer Society, and the IEEE Signal Processing Society. He has served as a reviewer for Computers & Graphics, the Journal of Visual Communication and Image Representation, the IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), the IEEE Transactions on Information Forensics and Security (TIFS), the European Signal Processing Conference (EUSIPCO 2018), the IEEE International Workshop on Information Forensics and Security (WIFS 2018), the IEEE CVPR Workshop on Media Forensics (CVPRW 2019), the IEEE Advanced Video and Signal-based Surveillance (AVSS 2019), and the IEEE Winter Conference on Applications of Computer Vision (WACV 2020).