Universidade Estadual de Campinas
Instituto de Computação

Waldir Rodrigues de Almeida

Data-driven face presentation-attack detection in mobile devices

Detecção de ataques de apresentação por faces em dispositivos móveis

CAMPINAS
2018

Waldir Rodrigues de Almeida

Data-driven face presentation-attack detection in mobile devices

Detecção de ataques de apresentação por faces em dispositivos móveis

Dissertação apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Thesis presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Computer Science.

Supervisor/Orientador: Prof. Dr. Anderson de Rezende Rocha
Co-supervisor/Coorientadora: Dra. Fernanda Alcântara Andaló

Este exemplar corresponde à versão final da Dissertação defendida por Waldir Rodrigues de Almeida e orientada pelo Prof. Dr. Anderson de Rezende Rocha.

CAMPINAS
2018

Agência(s) de fomento e nº(s) de processo(s): Não se aplica.
ORCID: https://orcid.org/0000-0002-5848-5560

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

AL64d

Almeida, Waldir Rodrigues de, 1990-
Data-driven face presentation-attack detection in mobile devices / Waldir Rodrigues de Almeida. – Campinas, SP : [s.n.], 2018.

Orientador: Anderson de Rezende Rocha.
Coorientador: Fernanda Alcântara Andaló.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Aprendizado de máquina. 2. Redes neurais (Computação). 3. Reconhecimento facial (Computação). 4. Biometria. 5. Dispositivos móveis. I. Rocha, Anderson de Rezende, 1980-. II. Andaló, Fernanda Alcântara, 1981-. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: Detecção de ataques de apresentação por faces em dispositivos móveis
Palavras-chave em inglês:
Machine learning
Neural networks (Computer science)
Human face recognition (Computer science)
Biometry
Mobile devices
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Anderson de Rezende Rocha [Orientador]
Adam Czajka
Hélio Pedrini
Data de defesa: 25-01-2018
Programa de Pós-Graduação: Ciência da Computação

Universidade Estadual de Campinas
Instituto de Computação

Waldir Rodrigues de Almeida

Data-driven face presentation-attack detection in mobile devices

Detecção de ataques de apresentação por faces em dispositivos móveis

Banca Examinadora:

- Prof. Dr. Anderson de Rezende Rocha (Supervisor)
  Instituto de Computação - UNICAMP
- Prof. Dr. Adam Czajka
  CSE - University of Notre Dame
- Prof. Dr. Hélio Pedrini
  Instituto de Computação - UNICAMP

A ata da defesa com as respectivas assinaturas dos membros da banca encontra-se no processo de vida acadêmica do aluno.

Campinas, 25 de janeiro de 2018

The purpose of computing is insight, not numbers.

(Richard Hamming)

Acknowledgements

During a long-term work like this, one invariably relies on many people, and the way that leads to it is also paved by many previous experiences. First and foremost, I would like to thank my family. Although my parents had never stepped foot near a university, when I showed interest in higher education, they immediately encouraged me. When I had to move to an expensive city, they gave me strength and supported me financially, without ever expecting anything in return. Father, mother, and sister, you are my safe haven – thank you for always supporting and believing in me! I truly believe I owe most of what I achieved to you.

Speaking of education, Brazil is a beautiful and unique country, but whether someone who only had access to its public primary and secondary school systems is able to realize his or her potential, is often a matter of luck. Therefore, I am eternally indebted to two very special teachers I had during high school: João Calixto Garcia and João Garcia Junior. Besides keeping a very high teaching standard in an environment where very few seemed to care, they also used to give free extra Mathematics and Physics classes, on the weekends. You were my role models, and you silently inspired me to dream of one day attending a top university, and to take responsibility for that dream.

When I was just starting to pursue my Master's Degree, I was lucky to meet my girlfriend and best friend, Paola, and though we stayed physically separated for most of that time, she was always there to give me meaning, and to keep me motivated and sane. Paola, you make me a better person, and no words could describe what you mean to me. I am also grateful to all unmentioned old and new friends, who in one way or another contributed to my personal growth. In particular, I am fortunate to have met and shared a home with some incredible people, from the very beginning of my undergraduate years. Julián Ávila, Julián Escobar, Alejandro, I will deeply miss our late night discussions!

This research was done in close collaboration with many people. My supervisors, Prof. Dr. Anderson Rocha, and Dr. Fernanda Andaló, are among the hardest-working people I ever met, and their achievements speak for themselves. Thank you for giving me the right amount of freedom and direction, and for being always available. I also thank Prof. Dr. Ricardo Torres and Prof. Dr. Jacques Wainer for the helpful pieces of advice during our meetings. Prof. Anderson and Prof. Ricardo have accompanied me since I was an undergraduate student, and I am indebted to them for many of the opportunities I had in the last years. To my friends and colleagues from the BioLive project, Rafael, William, and Gabriel, I already miss our spontaneous discussions about work and beyond, and I hope we can keep in touch. You certainly contributed in many ways to this work.

Finally, we would like to thank Motorola for the financial support. In particular, we also thank Benicio Goulart, Thiago Resek, Guilherme Megeto (from the Eldorado Institute), and everyone else who were in direct contact with us. The dataset collected as part of this work relied on many volunteers, to whom we are very grateful.

Resumo

Com o crescimento e popularização de tecnologias de autenticação biométrica, tais como aquelas baseadas em reconhecimento facial, aumenta-se também a motivação para se explorar ataques em nível de sensor de captura ameaçando a eficácia dessas aplicações em cenários reais. Um desses ataques se dá quando um impostor, desejando destravar um celular alheio, busca enganar o sistema de reconhecimento facial desse dispositivo apresentando a ele uma foto do usuário alvo. Neste trabalho, estuda-se o problema de detecção automática de ataques de apresentação ao reconhecimento facial em dispositivos móveis, considerando o caso de uso de destravamento rápido e as limitações desses dispositivos. Não se assume o uso de sensores adicionais, ou intervenção consciente do usuário, dependendo apenas da imagem capturada pela câmera frontal em todos os processos de decisão. Contribuições foram feitas em relação a diferentes aspectos do problema. Primeiro, foi coletada uma base de dados de ataques de apresentação chamada RECOD-MPAD, que foi especificamente projetada para o cenário alvo, possuindo variações realistas de iluminação, incluindo sessões ao ar livre e de baixa luminosidade, ao contrário das bases públicas disponíveis atualmente. Em seguida, para enriquecer o entendimento do que se pode esperar de métodos baseados puramente em software, adota-se uma abordagem em que as características determinantes para o problema são aprendidas diretamente dos dados a partir de redes convolucionais, diferenciando-se de abordagens tradicionais baseadas em conhecimentos específicos de aspectos do problema. São propostas três diferentes formas de treinamento da rede convolucional profunda desenvolvida para detectar ataques de apresentação: treinamento com faces inteiras e alinhadas, treinamento com patches (regiões de interesse) de resolução variável, e treinamento com uma função objetivo projetada especificamente para o problema. Usando uma arquitetura leve como núcleo da nossa rede, certifica-se que a solução desenvolvida pode ser executada diretamente em celulares disponíveis no mercado no ano de 2017. Adicionalmente, é feita uma análise que considera protocolos inter-fatores e disjuntos de usuário, destacando-se alguns dos problemas com bases de dados e abordagens atuais. Experimentos no benchmark OULU-NPU, proposto recentemente e usado em uma competição internacional, sugerem que os métodos propostos se comparam favoravelmente ao estado da arte, e estariam entre os melhores na competição, mesmo com a condição de pouco uso de memória e recursos computacionais limitados. Finalmente, para melhor adaptar a solução a cada usuário, propõe-se uma forma efetiva de usar uma galeria de dados do usuário para adaptar os modelos ao usuário e ao dispositivo usado, aumentando sua eficácia no cenário operacional.

Abstract

With the widespread use of biometric authentication systems, such as those based on face recognition, comes the exploitation of simple attacks at the sensor level that can undermine the effectiveness of these technologies in real-world setups. One example of such attack takes place when an impostor, aiming at unlocking someone else's smartphone, deceives the device's built-in face recognition system by presenting a printed image of the genuine user's face. In this work, we study the problem of automatically detecting presentation attacks against face authentication methods in mobile devices, considering the use-case of fast device unlocking and hardware constraints of such devices. We do not assume the existence of any extra sensors or user intervention, relying only on the image captured by the device's frontal camera. Our contributions lie on multiple aspects of the problem. Firstly, we collect RECOD-MPAD, a new presentation-attack dataset that is tailored to the mobile-device setup, and is built to have real-world variations in lighting, including outdoors and low-light sessions, in contrast to existing public datasets. Secondly, to enrich the understanding of how far we can go with purely software-based methods when tackling this problem, we adopt a solely data-driven approach – differently from handcrafted methods in prior art that focus on specific aspects of the problem – and propose three different ways of training a deep convolutional neural network to detect presentation attacks: training with aligned faces, training with multi-resolution patches, and training with a multi-objective loss function crafted specifically to the problem. By using a lightweight architecture as the core of our network, we ensure that our solution can be efficiently embedded in modern smartphones in the market at the year of 2017. Additionally, we provide a careful analysis that considers several user-disjoint and cross-factor protocols, highlighting some of the problems with current datasets and approaches. Experiments with the OULU-NPU benchmark, which was used recently in an international competition, suggest that our methods are among the top performing ones. Finally, to further enhance the model's efficacy and discriminability in the target setup of user authentication for mobile devices, we propose a method that leverages the available gallery of user data in the device and adapts the method decision-making process to the user's and device's own characteristics.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

In this work, we study the problem of automatically detecting presentation attacks against biometric face recognition systems in modern mobile devices, such as smartphones equipped with a camera. This chapter introduces the main concepts motivating the problem and presents our constraints and goals. Section 1.1 introduces Biometrics and Face Authentication in the context of mobile devices. Section 1.2 explains what presentation attacks are, and how they arise as a key security problem in face-recognition technology. Finally, Sections 1.3, 1.4, and 1.5 summarize our constraints, research questions, and contributions.

1.1 Biometric authentication in mobile devices

Biometrics, as the name implies, is an area of study that is concerned with measuring or characterizing individual living organisms. In the more narrower sense of biometric authentication, we wish to study how to use physiological or behavioral traits of a human individual to verify his or her identity. Examples of so called biometric modalities include but are not limited to fingerprint, face, and iris, as physiological traits, but also gait analysis, keystroke dynamics, and voice, as examples of behavioral traits [34].

The relevance of biometric authentication

Humans have always naturally identified other humans by their face, or voice. Biometrics as we know it emerged in the late 19^{th} century, when the distinctiveness of fingerprints was discovered, and they started being collected to identify criminals. With the evolution of computing and sensors, new forms of robust and automatic biometric recognition systems based on face, voice, and iris, among others, are being deployed.

More recently, users often need to secure the privacy of their digital data, secure access to their bank account or digital wallets, or prove their identity for claiming some other kind of service. In most of these applications, biometrics has risen as an alternative for knowledge-based methods of authentication, such as alphanumeric passwords or unlock patterns. Passwords are potentially very secure when used correctly, but this comes at the price of inconvenience, requiring users to memorize long strings of characters, which is not something humans are good at. Biometric authentication promises to eliminate the

need for such methods by using the individual itself as a "living password": one's identity is verified by who she is, instead of what she knows or what she owns.

A typical biometric authentication or verification system might look like the one depicted in Figure 1.1. The main components involved in data acquisition and processing are a biometric sensor, a feature extractor, and a feature comparison or matching module. Additionally, it is assumed that the user already registered his biometric data, which is stored in a *gallery* during the so-called enrollment phase. In the face-authentication case, the biometric sensor is a camera that captures an image of the user's face. The feature extractor is any method or algorithm that transforms image data into a more discriminative representation. Vectors stored in the gallery are also called templates. Finally, feature comparison could be as simple as a vector-similarity metric that compares the incoming user vector (probe) to the gallery, or a complex pattern classifier trained on many feature vectors. The output of the depicted authentication phase is a yes-or-no answer to whether the data is recognized as belonging to the enrolled user.

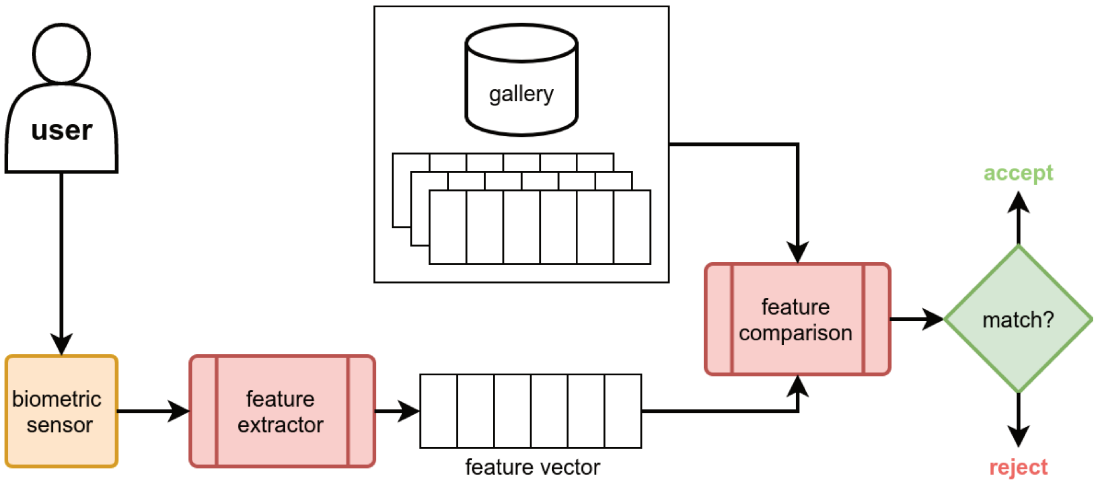


Figure 1.1: A generic biometric authentication system.

Effective biometric authentication assumes that users can be uniquely identified by examining the chosen trait, and that the system can capture the patterns asserting this uniqueness. Not all traits are universal or sufficiently discriminative for some security requirements, but systems based on machine learning and novel sensors are improving every day.

Why use face authentication in mobile devices?

In this work, we focus on the use case of user authentication in mobile devices through face recognition. More specifically, we consider the problem of secure user authentication for device unlocking.

The motivation for controlling access to a mobile device is clear. Smartphones have become so popular that they are almost an extension of the user's body and mind. Most people use their smartphone as their main medium of communication, storing conversational history, pictures, passwords, and other private data. Moreover, people are increasingly relying on their device for money transactions. In summary, as much as these

devices are useful, they must be secured, so that only the owner can access the data stored in it, even if its left unchecked on a table, lost, or stolen.

As already mentioned, although the traditional way to secure access to such devices has been through passwords or unlock patterns, these methods are inconvenient, in that users have to memorize digits or patterns, and make a concentrated effort every time they wish to gain access. Moreover, these passwords and patterns can be stolen by simply paying attention to what the user is typing. Mainly for those reasons, fingerprint and face recognition, or even iris recognition, are taking the lead as the preferred authentication method in mobile devices.

But which biometric trait should be used? Fingerprint is potentially more secure than face authentication, but requires its own specialized sensor. A fingerprint sensor usually occupies a significant area on the device surface, thus competing with screen size. Because of that, some prominent manufactures are moving away from this option. Iris or retina recognition, on the other hand, might also require additional sensors or a sufficiently powerful camera, good lighting conditions, and active user cooperation. Face recognition is thus the more convenient alternative, because it can be implemented using the stock front-facing camera present in most modern smartphones, and requires only that the user passively looks at the smartphone's screen, which is natural and not different from normal usage.

1.2 Presentation attacks and their detection

We have seen the value of face authentication in mobile devices. But even if the system is effective in discriminating genuine users from impostors, there remains the possibility of a malicious individual manipulating its components to gain access. That could happen at any point of the system, and could involve inserting or manipulating unprotected templates. In this work, however, our focus is in detecting a more direct type of attack that happens at the sensor level. It is a bigger threat in the sense that is does not require expert knowledge of the system to be executed, and is not easily detectable.

Vulnerability of face authentication to presentation attacks

A presentation attack (PA) against a biometric authentication system, also called a spoofing attack, is an attack that happens at the sensor level. In fingerprint authentication, for example, they would be perpetrated by using fake fingers that are created with synthetic materials to mimic the fingerprint of a target user. In face recognition, a PA can be made by simply showing the system an image of the target user, which requires little technical expertise, since face images are widely available on the Internet, and the attack procedure could be as simple as displaying the face image on an liquid-crystal display (LCD) monitor, in place of the actual user face.

Figure 1.2 illustrates how presentation attacks occur at the sensor level in a face authentication system. Instead of presenting the actual user face to the sensor, an impostor impersonates the target user by showing false biometric data: an image of the target user's face displayed on an *attack medium* or *instrument*.

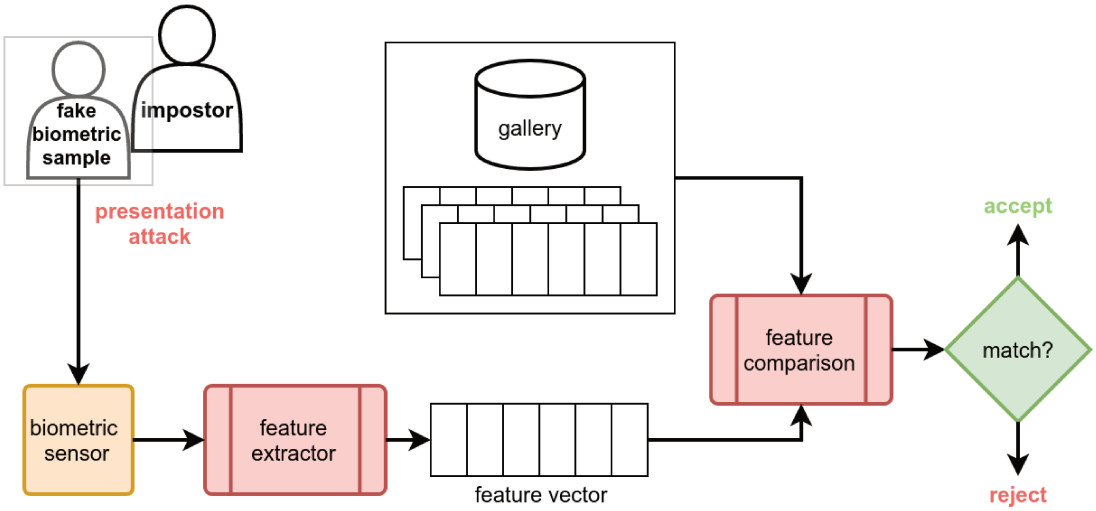


Figure 1.2: Illustration of a presentation attack in a face authentication system.

Types of attack

There are three main varieties of presentation attacks against face recognition systems: printed photo attacks, screen or display attacks, and 3D-mask attacks.

A printed photo attack consists of showing an image of the target user printed on a sheet of paper. This was the first type of attack to attract the attention of the research community [18, 75]. Variables that affect this type of attack include paper material and size, printer quality, color profiles, and ambient lighting. One important characteristic of printed photo attacks is that even though the attacker can move or fold the piece of paper, or even impersonate eye blinking by cutting holes in the paper, the actual image content is static.

Screen attacks are those in which the attack medium is a screen, such as an LCD monitor, or tablet. In contrast to printed photo attacks, the attacker can choose to display a video, instead of a static image. Attack quality depends on monitor size, pixel density, contrast levels, color reproduction, among other factors. Moreover, lighting comes mostly from the screen's backlight.

3D-mask attacks are a totally different class of attacks in which the attacker wears a mask that was built to resemble the facial features of the target user [27]. In contrast to both printed photo and screen attacks, the attack medium is an actual 3D object instead of a flat surface. Although relatively good-quality masks can be made from a few photos taken from different angles, the attack effort is greater than in the previous cases, which makes this kind of attack less likely to happen.

In this work, we focus on the detection of printed-photo and screen attacks. Other than being less common, we believe that the detection of 3D masks would require different methods, and therefore our focus is necessary to make the scope of our work reasonable.

Figure 1.3 shows examples of presentation attacks in comparison to a genuine authentication attempt. In this case, the biometric sensor was a webcam, and the figure shows individual frames that were captured by that device. Figure 1.3b shows a print attack, in which the attacker is holding a printed photograph in front of the sensor. Figure 1.3c

shows a screen attack performed with a tablet, which is replaying a video of the target user. An in-depth discussion of the characteristics of printed photo and screen attacks is provided in Chapter 2.

(a) Real access (b) Print attack (c) Screen attack

Figure 1.3: Examples of real access and attacks from the CASIA-FASD dataset [80].

1.3 Constraints of the mobile-device scenario

Modern mobile devices are powerful machines, but their processing speed and memory capacity are much more limited than those of a server equipped with multiple processing cores, dedicated graphics processing units (GPUs), and terabytes of RAM. Although it is possible to deploy machine learning models to a cloud server, and simply send captured user data and receive model predictions, this is undesirable for three main reasons: first, it is not reasonable to assume connectivity in every situation; second, sending full-resolution image data through a network can be slow and expensive; third, data privacy becomes an issue when raw user data is being transmitted.

On the other hand, in the main use-case of device unlocking, the user expects the authentication process to be as seamless and transparent as possible, since this is repeated at the start of each interaction.

Our constraints can be summarized in the following points:

- No connectivity is assumed: the whole pipeline should run on the user device.
- The complete pipeline must run in at most 1 second with modern smartphone hardware.
- Model should be small enough to fit in device memory without interfering with other applications. Ideally, memory footprint should not exceed 100 MB for a single prediction.
- The method should not depend on special user interaction, instead only requiring that the user looks at the frontal camera or screen.

1.4 Research questions

We guide our research by means of some investigative questions, considering the problem of software-based presentation attack detection in mobile devices, and the constraints specified in the previous section. Our guiding research questions are the following:

- Can we effectively and efficiently solve the problem of face presentation attack detection, in the context of mobile devices?
- Can a purely data-driven method outperform handcrafted methods in controlled benchmarks?
- In which cases software-based PAD methods are likely to fail in the real world?

1.5 Contributions

In summary, our contributions are the following:

- The proposal of three different ways of training deep convolutional neural networks to model and solve the problem in a purely data-driven way:
  - Training a lightweight architecture with aligned whole-face images;
  - Training with face patches of varying resolution, which reduces overfitting to user-specific characteristics, and promotes the learning of more robust representations that are not tied to a single scale;
  - Training with patches, and a loss function that more closely models the PAD objective, by promoting the compactness of intra-device genuine examples in the learned feature space.
- An extensive study of error cases, considering multiple factor-disjoint protocols.
- A simple but effective method for adapting a trained model by using a gallery of user data on the device.
- A novel face presentation-attack detection dataset, that is representative of our target scenario, with realistic illumination conditions, including indoor and outdoor sessions, in contrast to existing public datasets.

1.6 Thesis organization

Here we describe how the following text is organized. In Chapter 2, we give further context and motivation by discussing how software-based presentation-attack detection has been treated in prior art, giving an overview of existing public datasets and methods. Moving on to Chapter 3, we motivate our data-driven approach by introducing convolutional neural networks, and explaining the architecture that is at the core of our proposed methods for PAD in mobile devices. In Chapter 4, our three methods, which involve different

ways of looking at the problem and training a convolutional neural network to solve it, are described in detail. We also describe the procedure to adapt the trained model to a specific user and device. In Chapter 5, we explain datasets, metrics, and baseline methods used in our experiments. Finally, in Chapter 6 we present and discuss all experimental results. Chapter 7 concludes this work.

Chapter 2

Software-based Detection of Presentation Attacks

In this chapter, we start by looking at how real-access and attack images are created, discussing some general assumptions that are involved in software-based presentation-attack detection. Here, our definition of *software-based PAD* is similar to the one used in the existing literature [28]. It excludes methods that are based on additional sensors [42, 81, 26, 72], user-interaction schemes, as in challenge-response methods [38], or multi-modal biometrics [15]. Instead, we focus on methods that only depend on looking at the raw biometric sample: one or more RGB face images captured by a regular camera.

Continuing, Section 2.2 provides a detailed account of existing public datasets, and Section 2.3 presents a brief overview of some of the most relevant techniques in the literature. Finally, Section 2.4 exposes some of the problems with the state of the art, motivating our approach.

2.1 Image acquisition and attack clues

In order to understand how we could possibly differentiate real-access from attack images by looking only at pixel information, we need to understand how data is transformed before being ultimately acquired by the camera in the user device.

Figure 2.1 illustrates the differences in the image acquisition process for real-access and attack samples. While genuine or bona-fide samples are acquired as a single capture by directly photographing the authenticating user, in an attack scenario the biometric sensor actually recaptures a previously captured image of the user, which is displayed on an attack instrument. The presentation-attack detection problem thus consists in answering whether the captured biometric sample is genuine or not. In this context, our only resource is the biometric sample itself, and the hypothesis is that we can answer the question for many combinations of transforming factors, only by looking at raw pixels.

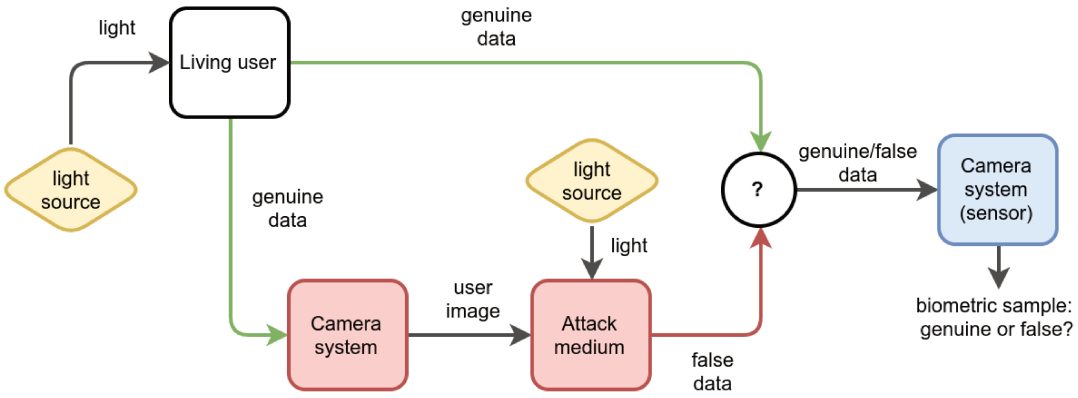


Figure 2.1: Image acquisition process for genuine and attack samples. Arrows represent the flow of data, while blocks represent the "actors" that transform it.

The crucial security concern is that an image or video of the rightful user captured by an arbitrary camera can later be obtained and used by a malicious user to make a false identity claim. This image could have been acquired under arbitrary illumination conditions, at an earlier date, and would be displayed on an attack medium during an attack.

We immediately notice that while a bona-fide sample is formed when a single camera-lens system captures photons directly reflected by a user face, attack images are formed by a more complex process. Firstly, an "attack camera" would have already captured and pre-processed an image of the target user face. Finally, when that image is displayed on the attack instrument, it is further modulated by the medium's own reproduction, geometric and reflectance characteristics.

Each part of the process changes the data to different degrees, but unfortunately it is not trivial to identify whether an image feature is due to the interfering attack camera or display medium, indicating an attack, or simply a normal variation in the user facial traits or lighting conditions during acquisition. All factors can vary arbitrarily and interact in seemingly unforeseeable ways.

In comparison to a similar genuine image, attacks often have different color distributions, due to limitations of the reproduction medium. Overall contrast is typically lower in printouts, due to soft focus and the influence of the light source on the flat paper surface. On the other hand, contrast is higher in most electronic displays, due to the strong backlight. Shadow and highlight areas are often rendered differently in both cases. Printouts can have visible printing defects, while low-quality liquid-crystal displays can suffer from varying brightness levels throughout the screen.

Finally, the resampling process often generates its own aliasing artifacts. A well-known example is the moiré pattern that appears when the sensor samples images containing fine-grained regular structures, such as the pixel grid in electronic displays. This only happens if the system can partially resolve the regular structure. Other regular artifacts can be caused by slow refresh rates in older displays, or low frame rates when replaying videos.

The discussion on possible attack clues will be continued in Section 2.3, in which we present an overview of existing software-based PAD methods.

2.2 Existing datasets

In this section, we introduce the most significant public datasets for face presentation-attack detection, with a focus on printed-photo and display attacks. As we shall see, some of the early datasets are not relevant anymore, given the evolution of cameras and other hardware, as well as the understanding of the problem itself. In particular, datasets and methods focusing on high-quality smartphone cameras are only now starting to appear.

The NUAA dataset [75] was the first face spoofing dataset to be released to the public, in 2010. It consisted only of printed-photograph attacks from 15 users, taken with a cheap webcam in different illumination conditions. Because of the small amount of identities, lack of a proper protocol, and low-quality of the acquisition device, it is no longer used.

Released in 2011, shortly after NUAA, the YALE-RECAPTURED database [60] was important for being the first public dataset to include liquid-crystal display (LCD) attacks. Nonetheless, it contains images from only 10 subjects, and the acquisition devices for real and attack samples differ.

In 2011, the Idiap Research Institute released PRINT-ATTACK [3], which was the first face spoofing dataset to include a more controlled protocol and 50 users. It contains 200 real access videos and 200 print attack videos. Still, considerable drawbacks are that it is based only on a single low-resolution webcam, and does not include display attacks. Its use in the *Competition on Counter Measures to 2-D Facial Spoofing Attacks* [14] showed that it is too simple to be used as a benchmark.

The CASIA-FASD dataset [80], released in 2012, aimed at solving some of the problems present in NUAA and PRINT-ATTACK, namely the lack of display attacks, and lack of multiple sensor devices. It was the first dataset to include 3 different cameras, which vary in their image-quality capabilities: a low-quality webcam, a slightly higher-quality webcam (referred as "normal quality"), and a high-quality *Sony NEX-5* camera, which records full-HD videos (1920×1080). The main drawback is that they only consider one indoors session with each camera, recording a total of 150 real-access videos from 50 subjects. Videos from the high-quality camera were used as the basis for display and printed-photo attacks. Videos were played back on a 1280×720 iPad screen. Photos were printed on A4 paper and used in two ways: warped-photo attack, in which the attacker introduces motion and some depth by slightly distorting paper corners; and cut-photo attack, in which the eye regions are cut out from the photographs, and the attacker uses his own eyes to simulate blinking. CASIA-FASD includes many different protocols that consider sensor types and attack types separately, as well as a an overall test considering all variations. One of the main critiques is that it does not include a development set, while the training set consists only of 20 users, which limits cross-validation alternatives.

The Idiap REPLAY-ATTACK dataset [16] was constructed in 2012 to extend PRINT-ATTACK by including display-based spoofs. It contains 200 real access videos from 50 users, taken in two different indoors sessions. One disadvantage is that it was built using only one low-resolution laptop camera as the biometric sensor, much like its predecessor. Screen attacks collected with an *iPhone 3GS* were displayed on the iPhone's screen, and attacks collected with a higher-quality compact camera were displayed on an iPad tablet screen. Additionally, print attacks on A4 paper were created by using a color laser

printer and the same compact camera. A total of 1,000 attacks were recaptured both by hand-holding the attack instrument in front of the sensor, and by using a fixed-support. One of the dataset's strong points at the time was the inclusion of an official protocol that divides the data into user-disjoint training, development, and testing subsets. This enabled the dataset to be used in *The 2nd competition on counter measures to 2D face spoofing attacks* [17]. In that competition, some teams exploited the image background to achieve an unrealistic perfect accuracy on the testing set. Despite the problems, it is still being used as one of the standard benchmarks for PAD algorithms.

The Unicamp Video-Attack Database (UVAD) [62] focuses on video attacks replayed on a screen. It is the first dataset to include hundreds of users and many different combinations of attack cameras and displays. Videos were recorded with 6 compact cameras in two different sessions: one indoors, and the other outdoor, with subjects being asked to stay still during the recording, as in other datasets. Despite containing videos recorded at a higher resolution than previous datasets, subjects were mostly recorded at a relatively far distance from the camera, which makes the effective resolution of the final cropped face low. The authors proposed many protocols that seek to assess the effect of using different combinations of monitors and cameras in an attack.

The MSU-MFSD [76] dataset was released in 2015, and includes videos taken with two acquisition devices: the frontal camera of a *Nexus 5* smartphone (720×480), and the webcam of a *MacBook Air* laptop (640×480). Its public version contains a total of 70 real access videos from 35 users, and 210 attacks from the same users. Video attacks are created with the back camera of an *iPhone 5S* smartphone and a *Canon 550D* digital single-lens reflex (DSLR) camera. The iPhone videos are played back on its screen, while the DSLR videos are displayed on an iPad tablet screen. The dataset also includes a printed-photo attack that is created with the 18-megapixel DSLR camera and printed on A3 paper, which potentially makes the attack harder to detect. The main issues with this dataset are that only half of its videos come from a smartphone, their resolution is low, and it only includes one illumination scenario.

REPLAY-MOBILE [20] is one of the recently-released datasets that aim to cover the mobile-device scenario. Its main merit is that it includes real access videos taken in 5 different illumination scenarios. Videos from 40 users were recorded in high resolution with one *LG-G4* smartphone and one *iPad Mini 2* tablet. It includes screen attacks with a Philips 227ELH monitor, and hard-copy attacks printed on A4 matte paper using a color laser printer. Like most face-PAD datasets, videos are mostly static, which limits the usefulness of using multiple frames for training and evaluation, and makes it non-representative of real-world usage of smartphones. One additional issue is that attack videos are taken only in a more controlled session, having consistent color reflections and color casting artifacts, as well as a different background in comparison to real-access videos, all of which contribute to create an artificial separation between the classes.

In 2017, the OULU-NPU dataset [11] was released as a dataset focused on the mobile-device scenario, being one of the most complete to this day. It includes 6 different smartphone cameras, 3 sessions, 2 print attacks based on different color printers, and 2 print attacks based on 2 LCD monitors, for a total of 4,950 videos. It is interesting to notice that some of the frontal smartphone cameras used as acquisition device actually have vari-

able focus, with one of the cameras (*OPPO N3*) actually being a back-camera that can rotate to the front. Despite its merits, it is still based on the tradition of using sessions of limited illumination variation, and very static video recordings, which is not completely representative of real-world smartphone usage. OULU-NPU was used as a benchmark in the competition of generalized software-based face PAD in mobile scenarios [10], at the *International Joint Conference on Biometrics* (IJCB), in 2017.

Finally, in this work we introduce the RECOD Mobile Presentation-Attack Dataset (RECOD-MPAD), with the goal of providing a realistic training and evaluation dataset for the mobile-device scenario. In contrast to existing datasets, RECOD-MPAD is based on videos collected under varying illumination conditions that cover most real-world situations: direct sunlight, outdoors with diffuse lighting (shadow), indoors with top light, mixed-lighting indoors with natural lateral light, and low-light indoors with top-lights off. During each recording with the frontal camera of a smartphone, users were asked to hold the device as they would normally do, and to slowly rotate around their own axis. This has multiple purposes: firstly, it makes the illumination vary during the video, by changing the angle of the illuminant to the face; secondly, it makes the background change from one frame to another; thirdly, the slow rotation introduces slight but varying motion blur throughout the video, which changes the effective resolution from one frame to another, and is more akin to normal usage, in which the user and the camera can move in relation to one another. Another important characteristic is the inclusion of screen attacks with two very different electronic displays: a typical 17-inch monitor, and a 42-inch TV that can display real-life sized images.

More detailed information regarding protocols and the construction of RECOD-MPAD are given in Section 5.1.1. Table 2.1 puts in perspective some of the characteristics of the most relevant public datasets. RECOD-MPAD is the only dataset with dynamic sessions and protocols that are explicitly based on frames.

Table 2.1: Comparison of recent or actively-used public face presentation-attack datasets.

Dataset	Videos	Frames	Resolution	Users	Sensor devices	Sessions	Attacks
RECOD-MPAD	2,250	143,997	high	45	2 smartphones	5 sessions, **dynamic incl. outdoors**	2 printed 2 displays
OULU-NPU [11]	4,950	variable	high	55	6 smartphones	3 sessions, static only indoors	2 printed 2 displays
REPLAY-MOBILE [20]	1,190	variable	high	40	1 smartphone 1 tablet	5 sessions, static only indoors	2 printed 1 display
MSU-MFSD [76]	280	variable	low	35	1 webcam 1 smartphone	1 session, static only indoors	1 printed 2 displays
UVAD [62]	17,076	variable	low/variable	404	6 compact cameras	2 sessions, static incl. outdoors	7 displays (x 6 cameras)
CASIA [80]	600	variable	low, high	50	2 webcams 1 compact camera	1 session, static only indoors	2 printed 1 video
REPLAY-ATTACK [16]	1,200	variable	low	50	1 webcam	2 sessions, static only indoors	2 printed 2 displays

2.3 An overview of existing methods

Over the last years, many methods for software-based presentation attack detection have been proposed. The interest in the problem has grown significantly since the release of the

NUAA dataset and the advent of the first competitions. Because of that, it is no longer feasible to give an exhaustive analysis of all published methods. It is however noticeable that most methods are related, and tend to be based on common assumptions and feature descriptors. In this section, our goal is to give a general but brief overview of the existing literature.

2.3.1 Methods based on liveness detection or motion

In the context of face verification, liveness detection methods are those that seek to detect presentation attacks through evidence for lack of vitality in the captured face. These methods typically depend on motion information, and thus assume the sensor captures a short video, instead of static pictures. Because of that, we include most methods based on motion analysis, in general, in the same category.

The archetypal liveness detection method is eye-blink detection [55, 56], which can be effective if the attacker uses a photograph printed on a piece of paper. As one of the early ideas, PAD countermeasures based on eye-blink detection are easily circumvented by video replay attacks, or even by cutting holes in the hardcopy and using one's own eyes to simulate blinking [80]. A significant drawback is that it depends on the user actually blinking, which can take several seconds.

Another class of methods tries to detect subtle movements characteristic of a living human face, as opposed to static printed photographs, or even low-fidelity reproductions in replay attacks. Overall approaches include optical-flow estimation [6, 39] and motion magnification [8]. Other approaches are based on temporal extensions of low-level texture descriptors [24, 40].

Some methods take advantage of motion correlations between foreground and background, or other scenic clues [2, 56, 78]. This is specially likely to succeed if the attacker does not use a fixed-support when performing the attack with a printout or tablet display, for example, but would probably fail otherwise.

In summary, liveness and motion-based methods have the disadvantage of requiring a potentially long sequence of frames to make a single prediction. Even if we are willing to wait for a full video to be captured and analyzed, most of these methods can be circumvented by faking eye-blinks and carefully handling the attack instruments.

2.3.2 Methods based on physics or geometry

Face presentation attacks using photographs or videos typically present the forged user representation on a flat surface, which has different reflectance properties than a living face. Some PAD methods therefore try to detect this "flatness" or abnormal reflectance with physical or geometric motivations.

One of the early methods tried to capture depth information via structure-from-motion techniques [18]. Others propose to detect differences in motion between nose and ears areas via optical flow estimation [37], or by explicitly modeling 3D projective invariants [25]. Another possibility is to model local curvatures by using multiple images [44]. These methods typically require at least some user cooperation to succeed.

By assuming a simplified Lambertian model of reflectance, in which lighting is completely diffuse, it is also possible to model the interaction between the illuminant and the reflective surface to extract albedo and normal maps [75], which are then used as representations to help discriminate real-access from attack samples. Although the motivation is clear and the resulting model is elegant, lighting in the real-world is mixed and uncontrolled, so the basic assumptions do not hold in practice. Another option is to model the diffuse and specular components to try to separate the latter, which could emphasize characteristics of the attack medium surface [5].

2.3.3 Methods based on texture, noise analysis or image quality

A broad category of software-based PAD methods seek to detect artifacts left by the recapture process, or estimate degradation in overall image quality.

Texture characterization is typically motivated as a means of discriminating the intrinsic textural properties of attack instruments and living faces, but can also capture other types of high-frequency information. Most characterizations are based on variations of local-binary pattern (LBP) descriptors on grayscale images [51, 52, 16, 41]. Temporal extensions were also proposed [24, 40]. Although most of these methods were tested for grayscale images, recent proposals highlight the importance of using color information [12, 13]. Other methods use a combination of low-level local descriptors [68, 30]

Frequency-specific information can be captured by difference-of-gaussians (DoG) filtering [60, 80], or through Fourier analysis [48, 35]. A more global characterization that discards content information in static-content videos to analyze noise signatures is proposed in [62], while the same type of residual information is encoded as mid-level temporal representations in [61]. These methods are typically capable of capturing moiré patterns and other regularities that arise during the recapture process. Another explicit approach is presented in [59].

Methods based on low-level texture descriptors or high frequency information can be effective in detecting paper texture and noise patterns, as demonstrated by their overall popularity. Nonetheless, the effectiveness is extremely dependent on the exact acquisition conditions, and the capability of the camera resolving fine details. Moiré-like patterns are very strong clues for attacks, but are not always present, making countermeasures solely based on them unreliable.

Explicit attempts at capturing image distortion artifacts can be found in [76]. More recently, researchers have explored using generic image quality metrics directly [29, 4]. The metrics range from traditional mean-squared error to more recent structure similarity index (SSIM) and visual information fidelity (VIF). Since some of these metrics require a reference image, which is not available, the authors in [29] compare the probe image to an artificially degraded version of itself. The hypothesis is that the difference is greater between real-access images than between attack images, since the latter are already of lower-quality.

The assumption that presentation attacks are of lower quality is not always valid. Consider, for example, a source face image of high quality that is recaptured in a controlled and well-lit environment. Although it is an attack, the resulting image would be sharper

and less noisy than a real-access sample captured under unstable and low-light conditions. In summary, although under similar acquisition conditions attacks and real-access samples would potentially be separable by generic image quality metrics and statistics, existing algorithms do not take context into consideration, which makes them brittle in real-world scenarios.

2.3.4 Methods based on feature learning

In 2012, Krizhevsky et al. [43] trained a convolutional neural network CNN from end to end to win the ImageNet competition on large-scale image classification [66], surpassing handcrafted methods by a considerable margin. Since then, models based on CNNs have achieved state-of-the-art performance on many image recognition tasks. Data-driven methods like these, which receive raw pixels as input during training, and learn intermediate representations directly from data, are said to perform *representation* or *feature learning* [7].

Feature learning is underrepresented in the face PAD literature, despite the successes of deep neural networks in other visual tasks. Menotti et al. [53] were the first to systematically study the potential of convolutional neural networks for spoofing detection. They considered not only face PAD, but also fingerprint and iris spoofs, and explored two different strategies: filter optimization and architecture optimization. The first one is the standard way of training a neural network by using stochastic gradient descent and the backpropagation algorithm [65], for a fixed architecture. Architecture optimization seeks to find a suitable architecture, given a constrained search space [63, 21]. In this case, many simple architectures with random convolutional filters were randomly sampled and used as feature extractors to train a final linear classifier for the problem.

During the face PAD evaluation, they only reported results for the REPLAY-ATTACK and the 3DMAD mask dataset [27]. Interestingly, despite obtaining competitive results via architecture optimization, they found out that the optimized architectures not only could not be improved by having their parameters further adjusted with backpropagation, but that actually made them dramatically lose accuracy in the testing set. This could partially be attributed to insufficient hyperparameter tuning involved in SGD training.

Yang et al. [79] trained a CNN based on the *AlexNet* from Krizhevsky et al. [43], using an SVM classifier at the end. During pre-processing, they experimented with a few face-centered regions, including tighter face crops, and regions showing more background. They reported promising results on both CASIA and REPLAY-ATTACK, but the best pre-processing configuration was different in each case. As one of their conclusions, they highlight the differences in background between the two datasets, which makes evident that the network learned to exploit capture biases when too much background was used in training.

The method by Li et al. [49] consists in extracting features from early layers of pre-trained CNNs, namely AlexNet and VGG-Face [70]. Dimensionality is reduced with Principal Component Analysis, and the classification model is an SVM classifier. Experiments were conducted with REPLAY-ATTACK and CASIA, and the results on the latter were similar to the ones obtained by Yang et al. [79].

Patel et al. [58] also experimented with training deep CNNs using aligned faces and the whole frame as input, but they considered only architectures that are unfeasible to be deployed on smartphones. The choice of training with whole frames is dubious, since it is strongly dependent on the dataset. The final system consists of a fusion scheme involving the output of the CNN, and an eye-blink detector.

We conclude by highlighting that our approach falls into this category. In contrast to existing methods, we are specifically concerned with the mobile-device scenario. Critically, none of the reviewed work consider modern datasets for that scenario, and so far the strategies for using the available data and training the networks have been limited.

2.4 A critical look at the state of the art

Early methods for face PAD were mostly based on eye-blink detection and other motion clues, which require several frames to be acquired, and typically fail under video replay attacks or simple cut-photo attacks. The community then moved on to exploring potentially more generalizable clues based on texture description, but currently we are stuck in a situation in which most of these methods are based on the same low-level descriptors and simple classifiers, and yet they were shown to completely fail under more challenging cross-dataset protocols [23]. Other more recent methods also suffer from the same problem [61, 13, 76].

As much as available public datasets have been useful for comparing different approaches, and inspiring new research efforts, they are now mostly outdated, both in terms of available cameras and attacks, and in terms of methodology. The partial shift to cross-dataset evaluations has shown the limitations of methods and datasets alike. Only recently has the community started to address the specific constraints of mobile applications [76]. New datasets, such as OULU-NPU [11] and REPLAY-MOBILE [20] have appeared, with accompanying modern protocols, but they still have some of the same problems as other datasets, such as static sessions with low lighting variability.

Finally, efforts in applying deep learning, or other data-driven approaches to face PAD have been limited, with most solutions based on very similar aligned-face pre-processing and training strategies, and not taking into account our stated constraints. To the best of our knowledge, there are no rigorous studies of such methods considering modern protocols and mobile devices. It is in this context that we propose to study the problem in a purely data-driven way, hoping to gain insight into how far we can go with software-based methods in our constrained scenario.

Chapter 3

Deep Convolutional Neural Networks

In this chapter, we briefly introduce and conceptualize a class of machine learning models known as artificial neural networks. Section 3.1 discusses neural networks in general. Section 3.2 explains *convolutional neural networks*, which is a particular variation of the basic idea that is suitable for modeling learning problems involving images. Section 3.3 expands on the observation that these networks can be trained from raw pixel data to perform *end-to-end classification*. We conclude, in Section 3.4, by explaining the *SqueezeNet* architecture, which is the base of our methods.

3.1 Neural networks

Neural networks are multi-layered non-linear models that are based on a connectionist principle. One of the basic assumptions is that complex relationships in data can be modeled by stacking linear transformations followed by non-linear *activation functions*. In a very general way, we can express the function modeled by a multi-layered neural network as having the following form:

$$f(\mathbf{x}) = (\sigma_L \circ W_L \circ \sigma_{L-1} \circ W_{L-1} \circ \ldots \circ \sigma_1 \circ W_1)(\mathbf{x}). \tag{3.1}$$

Here, $W_i, i \in \{1, ..., L\}$, are real-numbered matrices defining linear transformations.[1] Each $\sigma_i, i \in \{1, ..., L\}$, is an element-wise non-linear *activation function*.[2] This is important, because otherwise the composition would be equivalent to a single linear transformation $W = W_L W_{L-1} \ldots W_1$. In fact, it was proven that even a conceptually simple network with two layers can approximate arbitrary functions under mild assumptions, crucially if the output at the first layer is a sufficiently large vector [22, 31].

By appropriately pre-defining the size of the output, so that the network outputs a vector of size C, we can interpret this vector as representing class likelihoods (or log-likelihoods), which could then be optimized to match the correct class of each input vector in a classification problem with C classes. In this context, the other important assumption

[1] In practice, instead of a linear transformation, we would have an affine transformation of the form $W\mathbf{x} + \mathbf{b}$. Alternatively, we can represent these by augmenting the input vector with a constant 1, and absorbing the bias vector into the weight matrix.

[2] Historically, the activation function was either a sigmoid or tanh, but more recently, the most popular one is the less smooth but non-saturating *Rectified Linear Unit (ReLU)*, which has the form $\max(0, x)$.

is that, if all individual transformations are differentiable, then meaningful weights can be learned by gradient-based optimization methods, such as stochastic gradient descent [45].

The actual learning procedure thus involves the calculation of gradients of the output with respect to all parameters in the network. An efficient algorithm for that, known as *backpropagation*, was popularized in the eighties, and is a direct application of the chain rule of derivatives [65]. The name stems from the fact that after a single forward pass, the error at the output is successively back-propagated to parameters in previous layers. The resulting gradient represents the direction in which each parameter should be modified to locally decrease the overall error at the end. Owing to the linear property of derivatives, the whole process is very modular, and is mostly automated by modern packages.

3.2 Convolutional architectures

Many extensions to the basic structure of neural networks have been proposed over the years. In the case where the input is an image, a successful architectural idea is that of a *convolutional neural network* (*CNN* or *ConvNet*) [46, 47], which was initially proposed in the context of handwritten-digit classification. *ConvNets* are now at the core of state-of-the-art methods in most image recognition problems [43, 74].

Images reside in very high-dimensional spaces. For example, a $3 \times 100 \times 100$ image can be viewed as a 30,000-dimensional vector. For a 8-bit representation, where each pixel would take values in [0, 255], there are 256^{30000} possible images. Fortunately, relevant images, natural images in particular, have some structure. Nearby pixels are highly correlated, and the same basic structures typically appear at different regions in the same image.

The vanilla network architecture previously defined is poorly-suited to explore these characteristics. If the output size of the first layer is 512, even this single layer would already have $30,000 \times 512 = 15,360,000$ weights. Most of these weights would be correlated, but would have to be learned independently, and if the input image would be spatially translated even by a single pixel, output could change drastically.

Convolutional networks solve the problem by substituting weight matrices with convolutional filter banks. At each layer L, an input of size $D_{L-1} \times H_{L-1} \times W_{L-1}$ is convolved with D_L filter volumes of size $D_{L-1} \times K_L \times K_L$, creating a volume of size $D_L \times H_L \times W_L$.[3] These values are then typically transformed by an activation function, such as $ReLU$ [43], as before. If the image is properly padded, and convolutions are implemented with a stride of 1 in each direction, the spatial dimension of the output remains constant, and each of the D_L output maps correspond to the filter responses at each spatial position. Filters can then be interpreted as feature detectors.

For example, for an input image of size $3 \times 100 \times 100$, and 64 filters of size $3 \times 7 \times 7$ at the first layer, the output would be 64 maps of size 100×100, corresponding to the activations of each filter. This process would then output $64 \times 100 \times 100$ values, but require only $64 \times 3 \times 3 \times 7 + 64 = 9,472$ weights and biases. Each filter could be specialized

[3] In theory, convolutional filters could still be interpreted as the old weight matrices, but with some groups of weights corresponding to different pixels constrained to share the same value. In practice, convolutions are a better abstraction, and yield better implementations.

at detecting edges at a certain orientation, corners, or color blobs, which are naturally occurring structures in images. The next layer of detectors would then build on the representations of this layer to detect more complex structures.

In order to increase the number of filters in subsequent layers, without dramatically increasing the number of parameters and computational requirements, it is common to downsample the spatial dimension after a certain number of layers. This is typically done either by max-pooling operations, or by using strided convolutions. Intuitively, it can also be argued that this encourages better translational invariance in the learned representations.

In recent years, many architectural modifications have been proposed, including different activation functions, new ways to combine convolutions of different sizes, and normalization operations in-between layers. Nonetheless, the main tendency has been to stack small 3×3 convolutions, downsample by a factor of 2 in each dimension after a few layers, and increase depth as much as possible [70, 73, 32].

3.3 End-to-end classification

The stacking of layers separated by non-linear activations can be interpreted as building a hierarchical representation, or a transformation of the original space to a space in which the final objective is more easily achievable. In the case of classification, the learning procedure uses labeled data to alter weights and biases, so that intermediate layers learn to successively transform the input space into a space in which classes are separable by a linear classifier.

Since weights and biases of intermediate layers and the classifier are jointly learned, we say that the model performs end-to-end classification. Most of the knowledge is in defining the architecture, the objective or loss function in relation to the input, and tuning hyperparameters. This is in contrast to other forms of machine learning, in which most of the work is in feature engineering, and the actual learned classification model can be as simple as a linear classifier on top of these fixed features.

In this work, we take the more *data-driven* approach of using deep neural networks to model and solve the problem. We believe this to be promising, given the lack of robust handcrafted models in the PAD literature, which suggests the problem is still poorly understood. Our choice lets us focus on definitions and basic principles, in order to gain greater insight into the problem, and understand why current approaches do not work satisfactorily.

3.4 Specific architectures

Since the success of the original AlexNet CNN for image classification [43], much concentrated effort has been put into finding new architectures that are either more accurate, or require less parameters and computation. The networks of the VGG family [70] and later GoogLeNet [73] showed in practice the benefits of depth in multi-layered CNNs. In contrast to previous networks, VGG CNNs are based only on 3×3 convolutions with

ReLU activations, and max pooling operations with a stride of 2. The intuition for using only 3×3 convolutions is that by stacking multiple layers we get the effect of the larger receptive field of larger kernels, with a smaller number of parameters, but with increased depth and expressiveness.

The process of finding or designing new architectures is non-trivial. Even if we limit our search space or design decisions to a few operations, such as predefined filter sizes, fixed activation functions, and a certain number of layers, there are still countless possibilities and permutations that must be tested exhaustively. Brute-force architecture search can yield interesting results [21], but it can only feasibly be done in a very limited search space. Moreover, this strategy was already explored for spoofing detection and simpler architectures in the work by Menotti et al. [53].

A standard approach, when considering convolutional neural networks for new visual classification problems, has been to start from an architecture that has proven its value by achieving competitive performance in the ImageNet benchmark [50, 77, 64]. This takes advantage of the observation that what works well for one visual task probably works well in another related task. In this work, we take inspiration from one of the versions of the SqueezeNet architecture [32], and use it as our core architecture for all experiments. This decision was taken for two reasons. Firstly, the resulting network is small and fast enough to be directly embedded in mobile devices, in contrast to other popular architectures such as VGG-16 [70, 57]. It has a fully-convolutional structure, which makes interpretation of results easier, and is appropriate for modeling our kind of problem, as we shall see. Finally, it was proven to be more accurate than AlexNet, which validates its potential for experimentation.

SqueezeNet

The SqueezeNet v1.1 architecture [32], which is the base of our methods, is illustrated in Figure 3.1. Next to each arrow is the shape of the resulting output tensor for a single input image of size $3 \times 227 \times 227$. For instance, the fire module named *fire4* receives as input 128 activation maps of spatial dimension 28×28, and outputs 256 maps of size 28×28.

Fire modules – illustrated in the top-right corner of Figure 3.1 – are similar to inception modules [73], and are the main building blocks in this architecture. They massively reduce the total number of parameters in 3×3 convolutions by first squeezing the channel dimension of the input tensor with 1×1 convolutions. For example, the squeezing layer in fire module *fire5* reduces the number of maps from 256 to $S = 32$, which means the following 3×3 convolutional layer has $128 \times 32 \times 3 \times 3 = 36,864$ weights instead of $128 \times 256 \times 3 \times 3 = 294,912$. In addition to the 3×3 convolutional layer with E_3 filters, another 1×1 convolutional layer with E_1 filters also receives the output of the squeezing layer, and helps recover information, at a low computational cost. The output of a fire module is then the concatenation at the channel axis of the $E_1 + E_3$ activation maps. Fire modules are thus parametrized by S, E_1, and E_3, with $S < E_1 + E_3$.

The complete architecture starts with a 3×3 convolutional layer producing 64 activation maps, and then stacks 8 fire modules, for a total of 17 layers with learnable

parameters at its core. Spatial dimension is reduced via max-pooling operations after the first convolutional layer, and after the second and fourth fire modules. Compensating for the reduced spatial dimension, feature dimension is gradually increased. The classification layer consists of a dropout operation [71, 43], to reduce overfitting, and a convolutional layer producing a number of feature maps that matches the number of classes in ImageNet. By averaging each class-specific map individually, the network also reduces the total number of parameters by eliminating the need for fully-connected layers.

The original SqueezeNet v1.1 has approximately 1.2 million parameters, which can be stored in less than 5MB of memory, making it suitable to be used with mobile devices. In comparison, AlexNet and VGG-16 have 60 and 138 million parameters, respectively. More details regarding memory footprint and running time are given in Section 6.7.

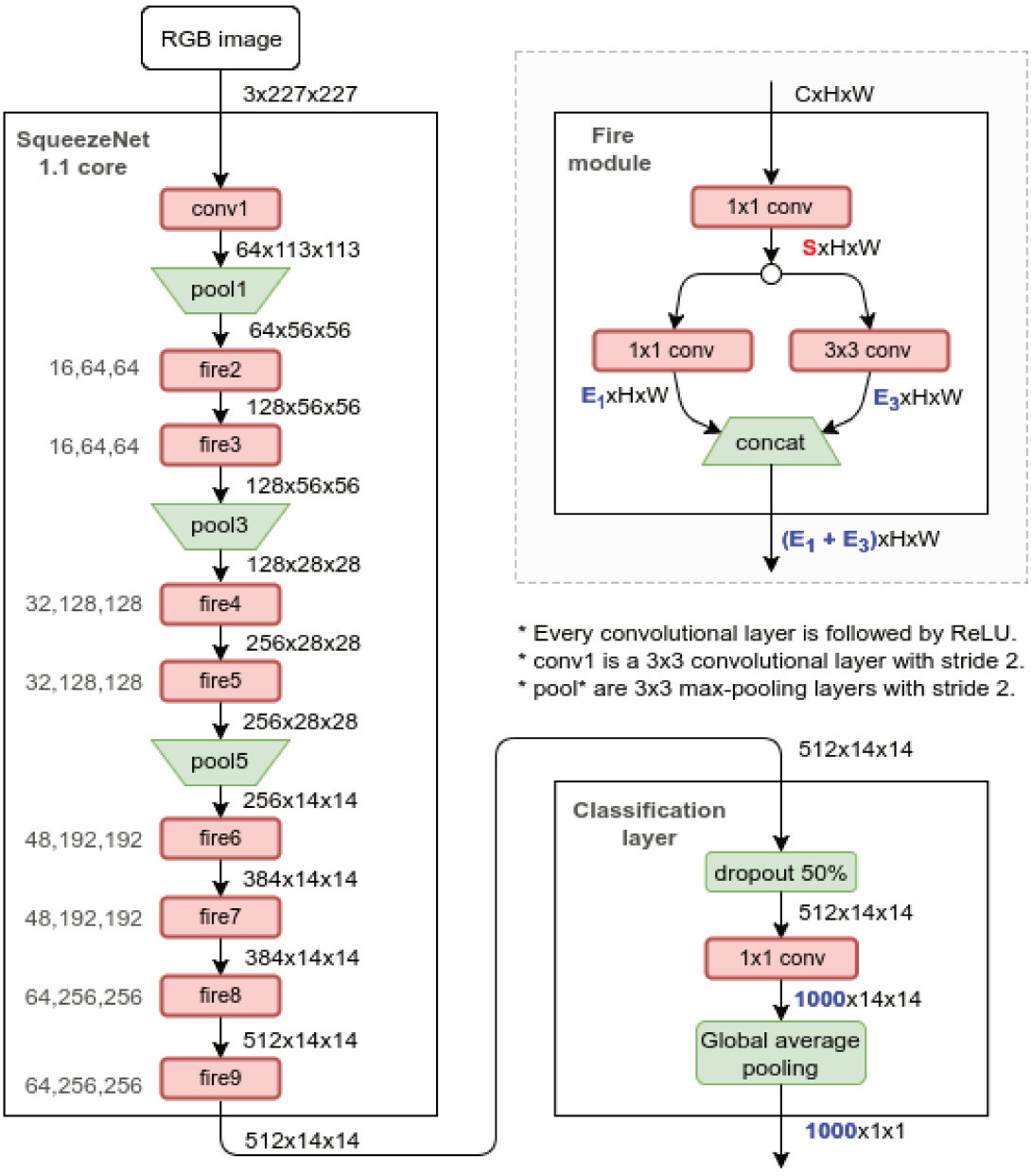


Figure 3.1: Original SqueezeNet v1.1 architecture [32], and generic micro-architectural details of a fire module.

Chapter 4

Proposed Methods

In this chapter, we present our approaches to presentation-attack detection in mobile devices. All three methods are based on training a convolutional neural network (CNN) to distinguish between real-access and attack images. They have the same deep stack of convolutional and max-pooling layers at their core, but differ either in what they see as input during training, or what they are optimizing at the end. This makes sense from a modeling perspective, because the interaction between the input and the optimization objective is what really defines the problem, driving the learning procedure.

In Method I, described in Section 4.1, we adapt the SqueezeNet architecture to consider presentation-attack detection, formulating the problem as 2-class classification, and using only aligned whole-face images for training the network. Method II, explained in Section 4.2, starts from the same modified architecture, but uses face patches of variable resolution during training, which reduces overfitting to user-specific characteristics, and promotes the learning of more robust representations that are not tied to a single scale. Finally, in Section 4.3, we use the same patches-based approach as in Method II for training the network, but propose a loss function that more closely models the PAD objective, promoting the compactness of intra-device genuine examples in the learned feature space.

4.1 Method I: CNN trained with whole-face regions

Our first method is based on training a deep convolutional neural network by using aligned whole-face images, which is the traditional input format in the pipeline of most algorithms published in the literature. In this case, however, the pipeline consists mostly of the whole multi-layered network, which is trained from end to end.

Training with aligned whole-face images can be justified as a means of reducing unnecessary variations during training and inference, putting the raw data in a predictable content-domain. This is similar in spirit to other forms of pre-processing, such as input domain normalization, which is often required by training algorithms. Although it is not obvious whether training with aligned whole-face images is the right pre-processing strategy for face PAD methods based on CNNs, it is certainly a valid first choice.

Architecture

The core architectural component for this and the other methods is the SqueezeNet network illustrated in Figure 3.1. As already mentioned, this decision was taken because we are interested in embedding the trained model in a mobile device with limited memory and processing power.

As this is a fully-convolutional network, all feature maps are flexible in size, including the input image. Nonetheless, we chose to keep the original shape of the spatial dimensions as 227×227. The spatial size of the raw cropped-face region in our scenario typically varies from about 300 pixels to about 550 pixels in each dimension. In face of this, keeping the original image size of 227×227 ensures that little detail is lost due to rescaling in most reasonable cases, while increasing it would also make the network too slow to train and be used in mobile devices. Most spoofing-detection pipelines in the literature pre-process images to a fixed size, typically varying from 64×64 to 256×256.

Pre-processing and data augmentation

We start from an aligned and square-cropped image of the face region, both in the training phase, and in the inference phase. The basic alignment process is described in details in Section 5.1.1, which introduces the RECOD-MPAD dataset. This alignment is done so as to ensure that we start from a region containing only the face. The original resolution is preserved by avoiding unnecessary rescaling at this stage. In practice, we found that the exact alignment does not significantly impact the performance of our methods.

During training, we read the RGB image, rescale the aligned face region to 256×256, and crop a central region of size 227×227. Before feeding the image to the network, we randomly flip the image horizontally with probability 0.5. This is a standard data augmentation strategy, when mirroring the image does not change the label semantics. On the other hand, we found that randomly cropping a 227×227 region from the rescaled image does not improve performance in comparison to center cropping, probably because the difference in content is not substantial and the fully-convolutional network is already robust to misalignments. Other data augmentation strategies that involve random photometric distortions and normalization are potentially too destructive for label information.

Before feeding the image into the network, we perform a simple pixel-wise transformation from the range $[0.0, 1.0]$ to the range $[-1.0, 1.0]$, by subtracting 0.5 from each pixel and then dividing by 0.5. Basic centering is common-place for getting meaningful gradients in the first iterations of training, especially if parameters are initialized randomly and the ReLU activation function is used [43, 70]. Researchers and practitioners typically subtract the mean pixel from the training set and divide by the corresponding standard deviation, but this is not strictly necessary for natural images, and creates an unnecessary dependence with the training data. In practice, we found that the described transformation does not affect the final accuracy significantly, but it helps to make the training procedure more stable.

Training details

Training is done via standard backpropagation [65]. Figure 4.1 illustrates the architecture and the training procedure for this method. We feed the network a pre-processed mini-batch of images containing faces and their labels.

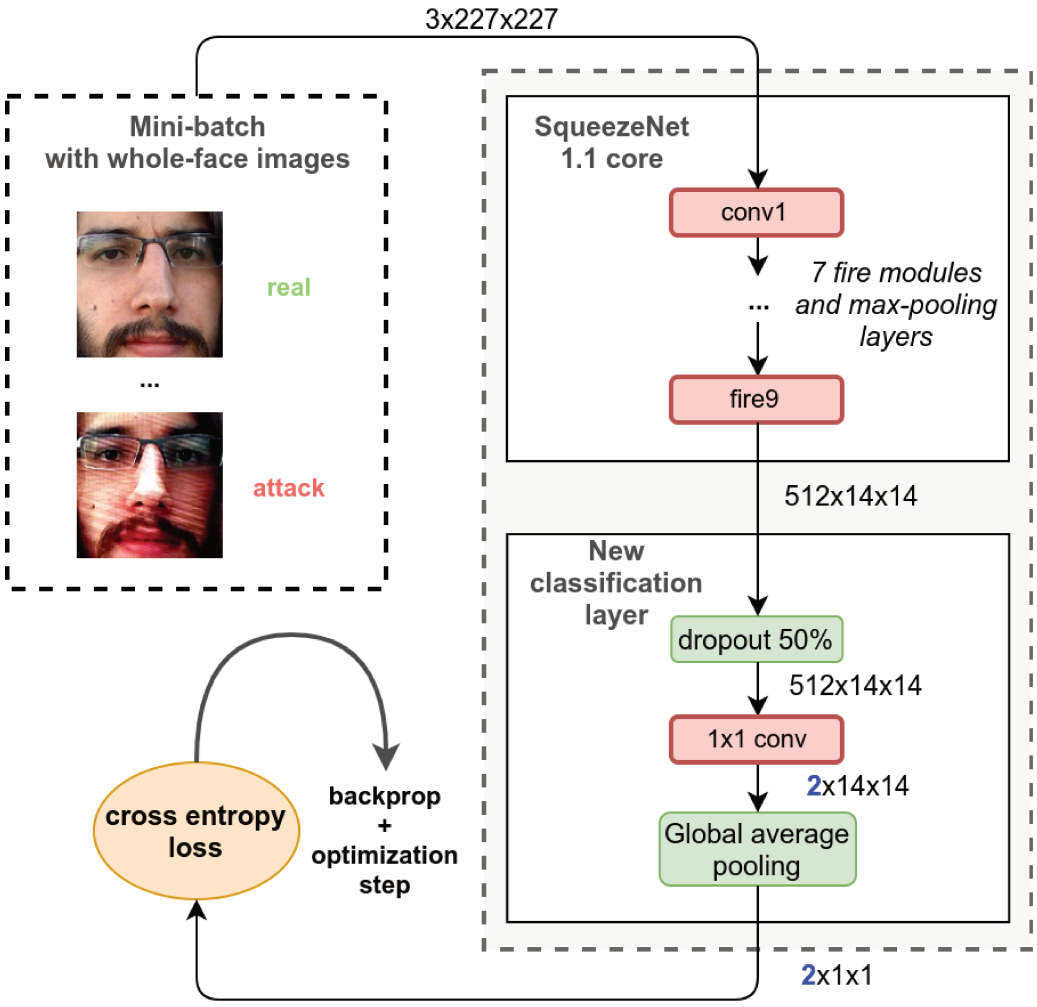


Figure 4.1: Network architecture and training procedure for Method I: *Whole-face CNN*. Note that, for illustration purposes, the diagram shows normal-looking images as they would appear before the centering transformation.

On each iteration, 64 images are randomly sampled with replacement from the training set. The probability of a single image being selected is set to be inversely proportional to the number of samples with its label in the training set, to account for class imbalance. Therefore, each mini-batch consists of roughly 32 samples with label *real*, and 32 samples with label *attack*. For purposes of validation and monitoring of the training procedure, we define an *epoch* as the number of training iterations required to process a number of images equal to the total number of images in the training set.

The 2-dimensional output corresponding to the two classes is used as input to a cross-entropy criterion, which is analogous to a traditional Softmax classifier. The resulting

function can be interpreted as normalizing the input vector into probabilities, and then measuring the mismatch between the predicted distribution and the expected distribution in which the mass is fully concentrated in the true label. In practice, we average over the whole mini-batch, giving the following expression, where $f_c(x)$ is the network output for class c and input x, and B is a mini-batch of training examples:

$$L(B) = \frac{1}{|B|} \sum_{(x,y)\in B} -\log\left(\frac{e^{f_y(x)}}{e^{f_0(x)} + e^{f_1(x)}}\right). \tag{4.1}$$

After computing the loss and intermediate activations, the gradient of the loss with respect to every adjustable parameter is computed via backpropagation. Finally, for the optimization step, we use the Adam optimizer [36], which is an adaptive optimizer based on stochastic gradient descent with momentum, requiring minimal hyperparameter tuning. All experiments were done with default Adam hyperparameters and a learning rate of 0.0001. As regularization, we add to the loss function an L2 penalty (weight decay) with weight 0.001. We found these hyperparameters to work well by monitoring the loss in the validation sets of the datasets we experimented with. In particular, increasing the learning rate by a factor of 10 makes the optimization diverge, while decreasing it by a factor of 10 makes optimization too slow, and increases the tendency to plateau at a suboptimal point.

As for parameter initialization, we start from pre-trained ImageNet weights[1] for the core part of the network, which is preferable to random initialization. For the classification layer, we initialize biases to 0.0, and weights from a normal distribution with mean 0.0 and standard deviation 0.01.

Inference

After the network is fully trained, it can be used to infer the label of new input images. In this case, pre-processing is mostly similar to the one during training. The detected face region is rescaled to 256×256, and then centrally cropped to 227×227. Each pixel in the aligned and cropped whole-face image is centered by subtracting 0.5 from it and dividing by 0.5, but in contrast to the training phase, no random mirroring is performed.

4.2 Method II: CNN trained with multi-resolution patches

In motivating a different approach to solve the problem, we start by asking the following questions:

> *Is a holistic view of the face really necessary to tell whether or not an image is a presentation attack?*

[1] Our implementation is completely based on PyTorch, and we use the pre-trained parameters made available with the library as of version 0.2.0.

Is learning from a fixed scale and level of detail sufficient to make the distinction, considering the low-level nature of the problem, and all possible variations of detail due to lighting, subject distance, noise, blur, and overall attack quality?

In light of these questions, our second proposed method differs from the first by modeling the problem as the task of distinguishing regions of arbitrary level of detail in attack images from regions of arbitrary level of detail in genuine images. We accomplish this during training by extracting patches of varying sizes from the full-resolution images, and only then rescaling them to fit the network input format. This approach is beneficial in many ways. Firstly, it increases the number of examples available for training, taking full advantage of the training data by not discarding information that would be lost by premature rescaling. By forcing the network to distinguish patches at different resolutions, its robustness to blur, adverse lighting, and unseen cameras is potentially increased. Finally, by not always showing the whole user face, the network is encouraged to not depend on user-specific characteristics, which could reduce overfitting.

Architecture

The architecture remains the same as in the first method. This is possible because we take advantage of its fully-convolutional nature. The network core consists mostly of stacked convolutional filters, which can be somewhat independent to scale. Different combination of these filters, acting as more advanced feature detectors, can naturally specialize or learn to adapt to the varying resolutions and feature scales. Moreover, the final classification layer implemented as global-average pooling acts as a parameterless aggregator of the final filter responses, and is thus invariant to locality in its input representation.

Pre-processing and data augmentation

Figure 4.2 illustrates the construction of a mini-batch in this method. Pre-processing for each image in a mini-batch starts by uniformly sampling a variable α from the interval $[0.08, 1]$, defining a percentage of the raw image area. The size of the cropped region is then $S = \sqrt{\alpha WH}$, where W and H are the width and height of the full-resolution whole-face image. In practice, the smallest possible patch corresponds to an area approximately equal to the region around one of the eyes in the original aligned image, regardless of the image size. The patch of size $S \times S$ in then cropped from randomly sampled top-left corners i, j. The remaining pre-processing and augmentation procedure is similar to the one in Method I, and includes rescaling the patch to 227×227, random mirroring, and centering.[2]

[2]Models for Methods II and III in this thesis were actually trained with an additional aspect-ratio augmentation, in which patches are sampled with a variable aspect ratio, before being rescaled to 227×227. The sampled rectangle is of size $W_P \times H_P$, with $W_P = \sqrt{\alpha r WH}$, $H_P = \sqrt{\alpha r^{-1} WH}$, and r sampled uniformly from $[0.75, 1.0]$. This type of purely geometric augmentation is cheaper than rotations, and has been used to train some models from the GoogLeNet/Inception family [73]. In terms of validation error, the models were not significantly different from training only with undistorted square patches, but the increased data variability could decrease test error. Thus, we see it as an optional addition.

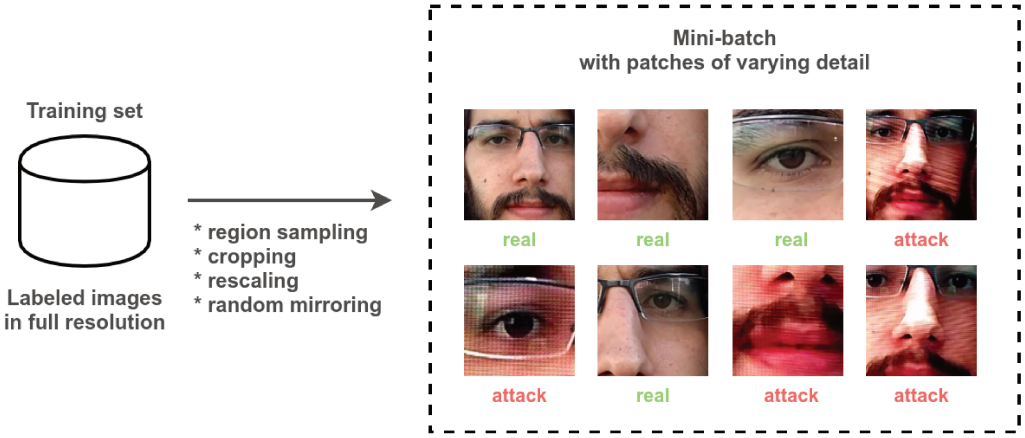


Figure 4.2: Construction of a mini-batch in Method II: *Multi-resolution patches CNN*.

As a consequence of this process, we effectively generate a much larger and variable number of examples from a single image in the dataset. Some patches will be closer to the native camera resolution and depict only part of the user face, while others will consist of most of the face, downscaled to the fixed input size. Different patches can emphasize different aspects of attack artifacts. As an effect, the trained network is expected to be more robust to variations in resolution. Moreover, the model must learn not to depend on certain combinations of facial features, which might be informative in the context of the users in the training dataset, but not in general, when dealing with previously unseen faces.

Training details

After constructing a mini-batch, training proceeds exactly as in Method I. Learning rate is set to 0.0001, weight decay to 0.001, and batch size is 64.

Inference

Crucially, because the network is fully-convolutional and is trained to be robust to variations in feature sizes, we can do fast and effective inference by using a single image. We simply feed the network the same whole-face image that is used in Method I. Alternatively, there could be considerable gains in accuracy by feeding multiple patches and averaging their response, but that would make the model too slow to run on mobile devices.

4.3 Method III: CNN trained with a multi-objective loss

When contemplating the problem of training models to be sensitive to a wide number of attack clues and sensor device specificities, we might ask ourselves the following questions:

> *Assuming data from multiple devices is available, what is the best way to account for their differences during training?*

Is it reasonable to assume that real samples from different devices should have similar characteristics?

Is pure binary classification the best way to model the problem?

How could we guide training to make the learned representations more likely to reflect actual differences in real versus attack samples, instead of dataset-depend noise that happens to separate real and attack samples from multiple devices?

Inspired by these questions, we propose another method, in which we reformulate the original problem by adding another term to the training objective, changing the way images are used during optimization. The goal is to encourage real samples *from a given device* to be more compactly located in intermediate feature spaces, but farther away from attack samples of the same device. We hypothesize that this would create better representations by not directly confounding information from different devices, as in traditional training strategies.

More specifically, we consider a latent representation $f_\ell(I)$ of the original input image I after it has been successively non-linearly transformed by ℓ layers. Consider a triplet of images *coming from the same device*, I_n, I_r, I_a, a real *anchor*, another real example, and an attack example, respectively, and let $n := f_\ell(I_n)$, $r := f_\ell(I_r)$, and $a := f_\ell(I_a)$, for short. Now, we can add the following loss function to the network:

$$L_{triplet}(\mathbf{n}, \mathbf{r}, \mathbf{a}) = \max\left(||\mathbf{n} - \mathbf{r}||_2^2 - ||\mathbf{n} - \mathbf{a}||_2^2 + m, 0\right), \tag{4.2}$$

where m is a margin hyperparameter that is fixed beforehand. By minimizing this loss, we are enforcing the notion that real samples from a given device should be closer to real samples of the same device sensor, but farther away from attack samples of the same device, up to a margin.

In theory, this kind of triplet loss could be used by itself to optimize an embedding to directly compare pairs of images during inference [67]. But since most benchmarks still do not provide a protocol that is based on using reference images, and we ultimately want the trained model to distinguish between arbitrary real-access images and attack images, we add the previously described cross-entropy loss to jointly enforce a classification objective:

$$L_{spoof}(N, R, A) = \kappa \max_{1 \le i \le T} \left[L_{triplet}(\mathbf{n_i}, \mathbf{r_i}, \mathbf{a_i})\right] + L_{class}(f(R \cup A)), \tag{4.3}$$

where N, R, and A are the sets of real anchors, non-anchor real images, and attack images in a mini-batch, respectively, and T is the number of triplets. All images in a mini-batch are sampled from a single device. $L_{class}(f(R \cup A))$ indicates that the cross entropy loss is computed only for the $2T$ non-anchor images. In practice, the embedding term is computed as the maximum triplet loss over triplets in a mini-batch. This can be interpreted as performing an online selection of hard triplets [67]. The hyperparameter κ controls the relative weight of the triplet loss term.

We can interpret each term as acting as a regularizer for the other objective. The classification loss encourages finding a single decision boundary that separates real-access

patterns from attack patterns in general, while the triplet loss encourages making intra-device real-access samples as compactly located as possible in the latent space, but farther away from attack samples of the same device.

Architecture

The core network architecture is the same as before, but we anticipate global-average pooling to directly reduce spatial correlations and the dimension of the activations of the *fire9* layer. This is the representation we use to compute the triplet loss. Following the 512-D embedding, we include the usual classification layer consisting of dropout and a linear mapping that generates class scores by optimizing the cross-entropy loss.

Figure 4.3 illustrates how the network is trained with the multi-objective loss. Mini-batches are built from triplets of images coming from the same device. Three different columns are formed, one with real-access anchors, one with real-access samples, and one with attack samples. The triplet loss is calculated for each triplet, and it encourages the compactness of a given device in the embedding layer, while driving attack samples away, up to a margin. In addition to that, the non-anchor columns are forwarded further into the classification layer, and their output is used to compute the usual cross-entropy loss. Our *spoof loss* is the weighted sum of these two loss components, with the triplet-loss component weighted by a parameter κ. The dashed lines indicate that weights are shared between the columns, i.e., we train a single network.

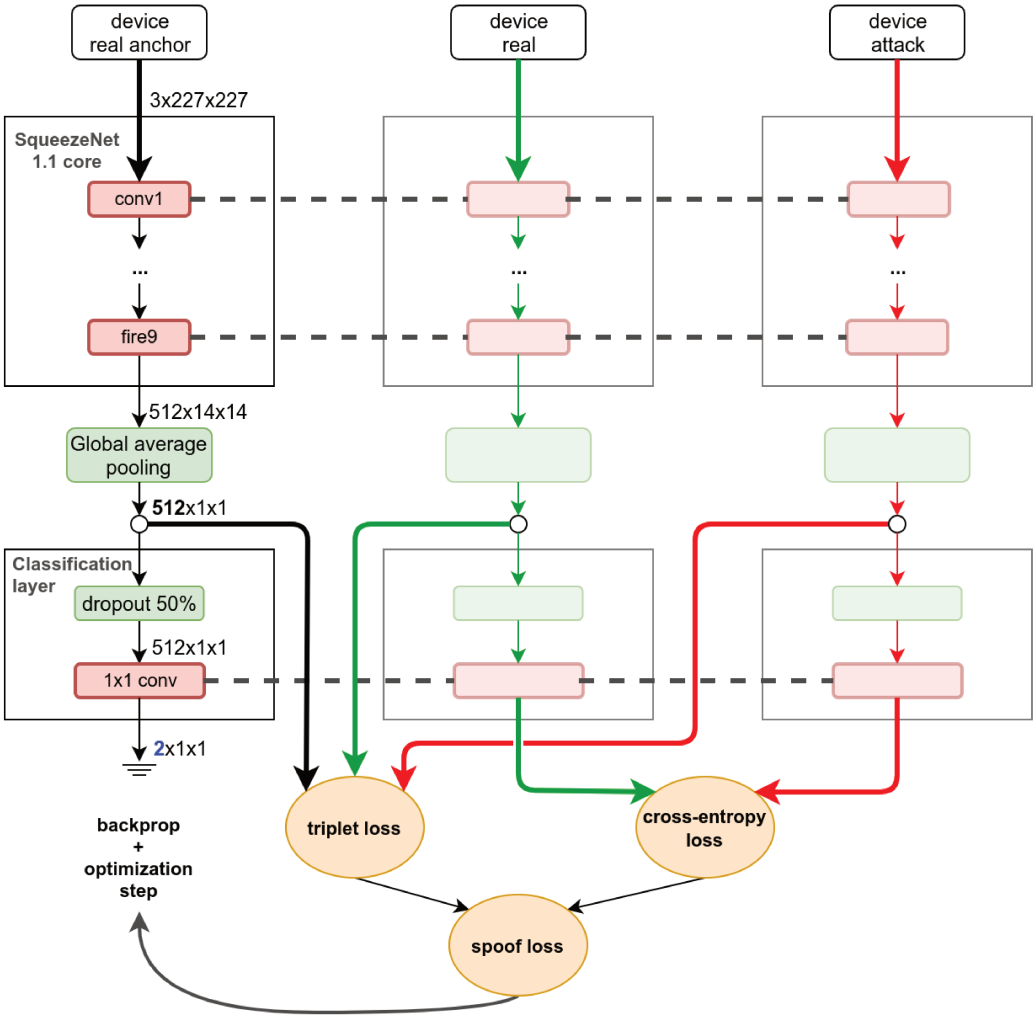


Figure 4.3: Architectural changes and the training procedure for Method III: *Spoof-loss CNN*.

Pre-processing and data augmentation

In this method, images from the training set are pre-processed into patches, exactly as in Method II.

Training details

As previously mentioned, we build mini-batches consisting of three columns of images from the same device: 64 real-access anchors, 64 real samples, and 64 attack samples. For the construction of each mini-batch, base images are sampled without replacement. Each triplet is passed through the network to calculate the aggregate maximum triplet loss component of the spoof loss, while the non-anchor samples are forwarded into the classification layer and used to compute the cross-entropy loss component. Gradients of the total *spoof loss* with respect to all parameters are then computed via backpropagation. Note that the triplet loss component does not influence the gradient at the classification layer.

The margin parameter m in Equation 4.2 was set to 1.0, and we found little value

in tweaking it, although, importantly, setting it to a large value can make optimization diverge in early iterations. The weighting parameter κ in the loss function was set to 0.05. Starting at 0.5 and successively reducing it by 0.1 and then 0.05, we found this value to be the largest that consistently does not make training diverge in the early stages.

The remaining details are analogous to the previous methods. We use the Adam optimizer, with a learning rate of 0.0001, and add an additional weight decay term with weight 0.001 to the loss function. Because Adam is an optimizer with momentum, even though we sample only from a single device on each iteration, gradients have a "memory", which reduces the need to accumulate them over multiple mini-batches before performing an optimization step.

Inference

Inference is done exactly as in other methods, i.e., we pass a single whole-face image through the network.

4.4 On-device user-specific adaptation

Previous sections described three different ways of training a convolutional neural network to solve the face PAD problem. In this section, we describe a method to further improve the effectiveness of these models in real-world situations, when they are deployed to mobile devices.

Classification models are trained with a finite training set, but are expected to work properly when presented with new data. Typically, if the operational data distribution is similar to the distribution of training data, models tend to behave well, but in practice there are no guarantees of generalization. More often than not, demands of the operational scenario are more specific. For example, in our case, the model will be deployed to a specific device, and will always be presented with images from the same user. In face of this, we propose one simple strategy to adapt the decision boundary to the specific characteristics of this user and sensor device.

Building a gallery of user scores

During normal operation, and possibly over the course of many days, the user will have successfully authenticated multiple times, in different lighting situations. Perhaps even his appearance will have changed, due to changing glasses, haircut, facial hair, etc. If we assume that after each successful authentication the final score (probability of attack) is stored in a user gallery, after some time we will have a representative distribution of the system's score for the real-access class. Alternatively, this gallery could be explicitly updated during enrollment sessions.

Threshold estimation

Our strategy takes advantage of a gallery G of user-specific real-access scores stored on the user device during normal operation. We assume that these scores are in the range $[0, 1]$, and that a higher score stands for higher likelihood of the input being an attack. We wish to find a minimal user-specific ψ, such that the false rejection rate FRR_ψ is bounded by a predefined value ϵ. The procedure is described in Algorithm 1.

Algorithm 1: User-specific threshold estimation.

Input: score gallery G, tolerated FRR ϵ, threshold increment Δ
Output: user-specific acceptance threshold ψ

```
ψ ← 0;
while CummulativeDistribution(G, ψ) < 1 − ε do
    ψ ← ψ + Δ;
end
return ψ
```

Function `CummulativeDistribution`(G, ψ) returns the cummulative distribution of scores in the gallery, from 0 to ψ. For example, if $\epsilon = 0.05$, the search starts from 0 and stops at the first threshold ψ for which no more than 5% of the real-access examples in the gallery would be rejected.

Final remarks

Despite its simplicity, the described procedure effectively tightens the decision boundary, so that the number of false rejections is controlled, while false acceptance errors are possibly reduced. In this way, a model is made more useful, in that it is adjusted to make the best prediction it can for attacks, constrained to a certain user-inconvenience level, represented by ϵ.

To see that performance can be improved, suppose that, at the default threshold, we observe 95 real-access attempts with scores no greater than 0.3, and 5 real-access attempts whose scores fall between 0.3 and 0.35. Moreover, suppose that from 10 attacks observed during the same period, 5 receive scores greater than 0.5, but 5 receive a score strictly between 0.4 and 0.5. In this case, a default threshold would induce an FRR of 0%, and a false-acceptance rate (FAR) of 50%. On the other hand, if the system had previously observed a similar distribution of real-access attempts, and learned a user-specific threshold of 0.3, the result for the new observations would be $FRR_{0.3} = 5\%$ and $FAR_{0.3} = 0\%$.

Chapter 5

Datasets and Experimental Methodology

Training and evaluation of methods based on machine learning require representative datasets and meaningful evaluation protocols. In Section 5.1, we describe details of the datasets used in our experiments. Section 5.2 discusses the metrics used in the evaluation protocols. Finally, in Section 5.3 we describe two baselines against which our methods will be compared.

5.1 Datasets

Building representative datasets is one of the hardest aspects of machine learning, being particularly hard in our case, in which biometric data is involved, and we cannot account for all possible attack configurations. In a sense, they serve as the problem definition, and are used both in training and evaluating models. Without a proper training set, we cannot train a useful model. And without a proper evaluation protocol, we cannot be sure if the trained model is useful.

We saw in Section 2.2 that most of the existing public datasets do not fully satisfy our requirements, given our constraints and problem domain. Because of that, we choose to focus on two datasets. First, in Section 5.1.1 we describe the construction process and the details of RECOD-MPAD, a dataset that we collected as part of this work. In Section 5.1.2, we describe OULU-NPU, which is another dataset that was recently released for the mobile-device scenario [11]. As we will see, it does not represent our use case as well as RECOD-MPAD, but is complementary to it. Moreover, its use as a competition benchmark in the year 2017 makes it particularly interesting.

5.1.1 RECOD-MPAD

In this section, we describe the processes of planning and collecting the RECOD Mobile Presentation-Attack Dataset (RECOD-MPAD), including the capture of real videos, re-capture of attack videos, and pre-processing. The main goal in creating this dataset was to have data that is truly representative of our fast mobile-device unlock scenario. Moreover, we wanted to cover as many illumination variations as possible, as this is something

that is lacking in public datasets. As such, it is the main benchmark for our methods. Because capturing biometric data and making it available involve ethical aspects, the project was sent to Unicamp's Institutional Review Board, being approved under protocol *CAAE 53035216.6.0000.5404*.

We started by selecting two main acquisition devices: a popular Moto G5 smartphone, released in 2017, and hereafter called device 1, and a Moto X Style XT1572 smartphone from late 2015, hereafter called device 2. These smartphones are equipped with modern frontal cameras that are quite different from one another. Both have fixed focus, but device 2 is sharper at a closer distance, and is in general much sharper than device 1. On the other hand, device 1 has a wider field of view, and is slighly brighter than device 2.

Collection of genuine videos

We carefully designed five illumination scenarios for capturing the real-access videos, which were also used as a basis for the recaptures. The five scenarios are the following:

- **Session 1:** Outdoors, with direct sunlight on a sunny day.
- **Session 2:** Outdoors, in a shadow, resulting in diffuse lighting.
- **Session 3:** Indoors, with artificial top light.
- **Session 4:** Indoors, with natural lateral light (window or door).
- **Session 5:** Indoors, with lights off (noisy).

Each session has its own characteristics. Figure 5.1 shows some representative examples, with cropped face regions. Frames in session 1 are almost noiseless because of the vast amount of available light, but tend to have strong specular reflections or suffer from lens flare, which are visually similar to artifacts found in some attacks. On the other hand, besides being darker, frames in session 5 are very noisy. Frames in session 2 are mostly well behaved, because of the diffuse lighting. Session 3 is the most similar to how other PAD datasets were collected. Session 4 represents a mix of direct natural light with diffuse reflections, potentially with more intra-session variations, due to our capture strategy.

Figure 5.1: RECOD-MPAD: variations between real-access sessions and acquisition devices. From left to right: sessions 1 to 5. Top row: device 1. Bottom row: device 2.

We instructed each user to hold the smartphone as he or she normally holds it, but to try to keep it frontal. We then instructed them to slowly rotate around themselves while capturing a video, so that a full rotation would take around 10 seconds. This ensured that we could get the maximum amount of variations in terms of illumination, motion and background, while keeping motion blur minimal. In total we captured 10 videos from 45 users, resulting in 450 real access videos. Figure 5.2 illustrates some of the variations that can be encountered in a single video.

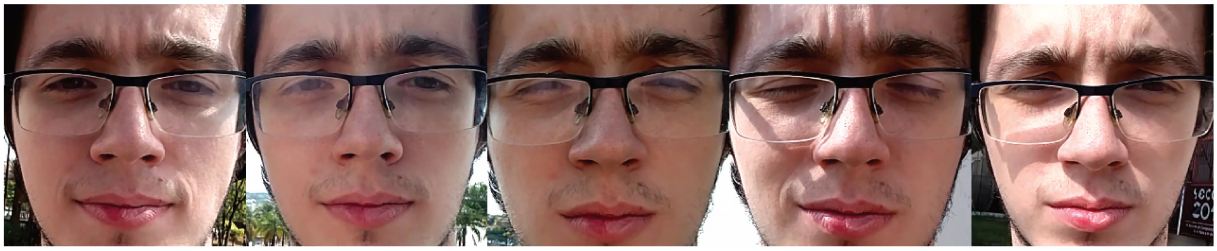


Figure 5.2: RECOD-MPAD: variations encountered in a single session.

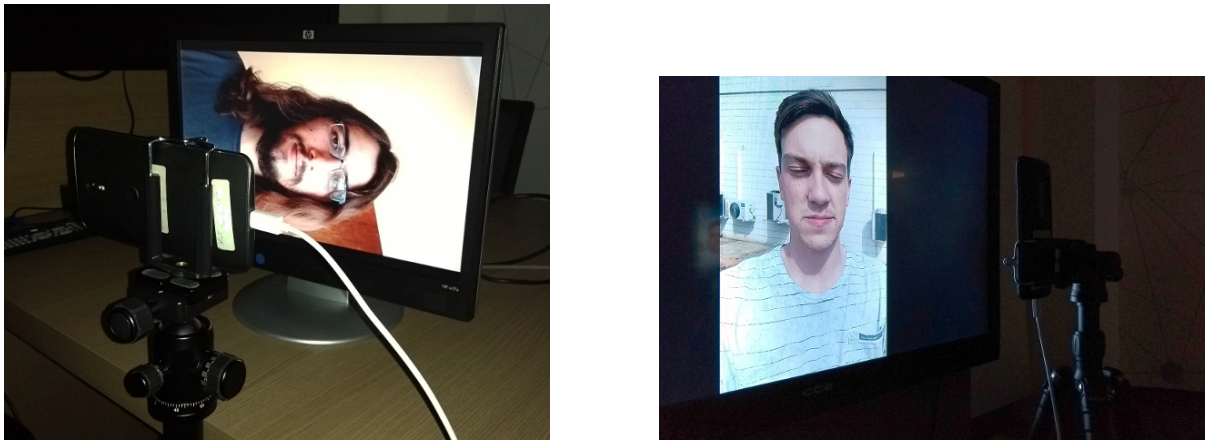
Recaptures

The recaptures were divided in two parts: display attacks, and printed-photo attacks. For the display attacks we chose two displays with different sizes: a CCE 42-inches full-HD TV (model LH42G), and a HP 17-inches desktop monitor (model w17e GV537A). Mainly because of the different sizes and pixel density, these displays are quite complementary.[1] The CCE TV can display images close to real-life size, with minimal luminance fall-off. Because of its size, the interaction with a fixed-size camera tends to result in a sharper image, simply because the camera can be placed farther away, at a proper focusing

[1]We also considered using a smaller tablet screen as one of the display attacks, but this was not included because it was impractical for us to collect more than two display attacks. In practice, the resulting images were visually similar to the recaptures from the 17-inch monitor.

distance. On the other hand, because of the lower pixel density, aliasing or moiring artifacts tend to be much more prominent. The smaller monitor results in blurrier images with darker corners, and is more similar to the displays used in other datasets.

Display attacks were performed with the smartphone on a steady tripod at a fixed distance from the monitors, except for small variations from one video to another. The distances were set to reduce aliasing artifacts, while covering the maximum possible video area. Lights were turned off to reduce reflections, and an opaque surface was held in front of the smartphone to eliminate reflections from its backlit screen on the attack screen. Each video captured with a given device was displayed on each monitor and recaptured with the same device. In total, we recaptured 900 videos. The setup is shown in Figure 5.3.



(a) HP 17" monitor. (b) Large CCE TV.

Figure 5.3: RECOD-MPAD: recapture setup.

Printed-photo attacks were created by extracting two frames from each of the original videos, roughly at 25% and 75% of the total duration. The images were then printed on high-quality A4-sized glossy paper, by using a professional BIZHUB C308 color laser printer. The first frame was then recaptured in one scenario with abundant natural diffuse light, while the second frame was recaptured in a darker living room also lit with diffuse natural light. The devices were set on a table with the camera pointing to the ceiling, while the attacker hand-held the hard copy. For about 8 to 10 seconds, the attacker started by holding the photograph close to the camera, while warping it and then pulling it away from the camera. The recapture finishes with the attacker slowly pushing the photograph towards the camera. This process was used to increase the number of variations between frames, since the photographs are static, and to simulate possible attack behavior, in the spirit of CASIA-FASD. In total, we generated another 900 attack videos. Figures 5.4 and 5.5 illustrate some of the differences between real-access and attack frames.

Figure 5.4: RECOD-MPAD: examples for acquisition device 1. From left to right: real, print 1, print 2, display 1, display 2. Top row: session 3. Bottom row: session 5.

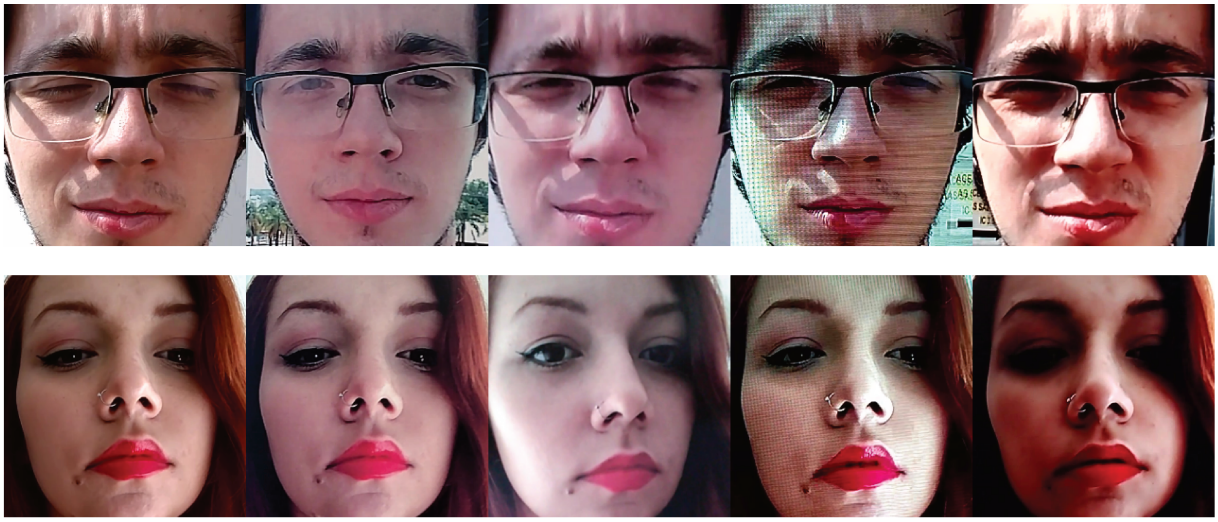


Figure 5.5: RECOD-MPAD: examples for acquisition device 2. From left to right: real, print 1, print 2, display 1, display 2. Top row: session 1. Bottom row: session 2.

Pre-processing

All videos had their audio removed and were cut so that their final duration is 8 seconds. For the official frame-based protocols, we extracted 64 equally-spaced frames from each video with maximum quality, which resulted in 144,000 frames. These frames were then fed into the face and landmark detector of the DLib toolkit[2], and localized eye centers were saved. For frames in which either face detection or landmark localization failed, we manually annotated eye centers using a custom *graphical user interface* tool. The final number of frames is 143,997, because 3 frames did not contain faces.

Using the locations of each eye, we proceeded to aligning and cropping each image to a square region by applying a similarity transform, so that the line connecting both

[2]http://dlib.net/

eye centers is made horizontal and they occupy a standard position in the cropped face region. The transformation closely preserves the original resolution by mapping each eye so that the distance between them is roughly 50% of the final crop size, which is defined as two times the original inter-eye distance. To reduce the introduction of artifacts during this phase, the final alignment is done through an interpolation of order 5 (bi-quintic).

All the code for pre-processing the videos in the dataset will be made available with the official release of the data.

5.1.2 OULU-NPU

The OULU-NPU dataset is a recently released public dataset for face-presentation detection in mobile devices [11]. It does not completely fit our scenario, since it is based only on 3 static indoor sessions. Its main merit is that it includes real-access and attack videos taken with 6 different smartphone cameras. More importantly, it was used in the *IJCB-2017 competition on face presentation-attack detection in mobile authentication scenarios*[3]. Because of that, we use it as an additional benchmark, which lets us compare our methods to the ones that were part of the competition.

In this section, we highlight some of its characteristics. Some of the details were already discussed in Section 2.2, and a more in-depth description can be found in the original paper [11]. Protocols are fully described in Section 6.3.

Collection of genuine videos

Real-access videos were captured with the front camera of 6 smartphones: *Samsung Galaxy S6 edge, HTC Desire EYE, MEIZU X5, ASUS Zenfone Selfie, Sony XPERIA C5 Ultra Dual, and OPPO N3*. All videos were captured in full-HD.

Most of these smartphone cameras are unusual, with only 2 smartphones having normal fixed-focus frontal cameras: *Samsung Galaxy S6 edge*, and *MEIZU X5*. *OPPO N3* actually has a single rotating camera, which is comparable to a normal high-quality back camera. The other 3 frontal cameras have auto-focusing capabilities. This is important, because focusing dramatically influences the characteristics of resulting images in close-distance recaptures. If a camera can properly focus on the attack surface, the result will be a more detailed depiction of the original scene, but this can also emphasize details on the attack surface and result in aliasing or moiring artifacts.

In general, for each camera and user, three 5-second indoors videos were captured. In the recordings, subjects and cameras remain relatively static, and the background does not change.

Recaptures

Attacks were based on photos and videos captured with the back-camera of a *Samsung Galaxy S6 Edge* smartphone. Videos were then displayed on two different monitors, a

[3]https://sites.google.com/site/faceantispoofing/home

19-inches desktop monitor, and a 13-inches laptop display. Photos were printed on glossy A3 paper using two different printers.

As in RECOD-MPAD, print attacks were created by holding the device in front of a fixed smartphone camera for about 5 seconds. Unlike RECOD-MPAD, movements are very small and there is no change in paper distance to the camera throughout the video. Since some of the cameras have variable focus, in some videos we can notice the effect of auto-focus hunting, effectively causing some variations in resolution during very short intervals.

Pre-processing

The official dataset comes with video files consisting of bundled JPEG images, and no audio. Accompanying each video is a text file with eye positions that were automatically localized with *DLib*, similarly to RECOD-MPAD. Some eye locations are missing.

For the purposes of this work, since videos are mostly static and frames are very similar, we extracted and kept only 1 in 7 available frames, which results in 17 to 21 frames per video, in most cases. We manually localized eyes, in case annotations were missing. The remaining pre-processing is exactly as the one for RECOD-MPAD, described in Section 5.1.1. We aligned and cropped face regions in a way that preserves the original resolution. This is the raw input to the algorithms.

In total, we have 96,661 frames from 4,950 videos, covering 55 users, 3 sessions, 6 sensor devices, 2 print attacks, and 2 display attacks. Each official protocol considers user-disjoint training, validation/development, and test sets, containing videos from 20, 15, and 20 users, respectively.

5.1.3 Summary

Table 5.1 summarizes the properties of the two datasets used in this work.

Table 5.1: Summary of the properties of datasets used in this work.

Property	**RECOD-MPAD**	**OULU-NPU**
Number of videos	2,250	4,950
Number of frames	143,997 (official protocol)	variable
Number of users	45	55
Smartphone cameras	2 (fixed focus)	6 (fixed or auto-focus)
Pixel resolution	1920×1080	1920×1080
Number of sessions	**2 outdoors**, 3 indoors	3 indoors
Session type	**dynamic**	static
Display attacks	**1 large**, 1 medium	2 medium
Print attacks	2 (**diff. recapture lighting**)	2 (**different printers**)

5.2 Evaluation metrics

In order to evaluate the efficacy of trained models, we use the standard metrics of the presentation-attack detection literature. For a given frame or video, the output of each algorithm is a score that represents its confidence that the input should be classified as *attack* as opposed to *real*. In this work, all scores can be interpreted as probabilities in the range $[0, 1]$.

False acceptance rate (FAR)

Let $f(x)$ be the predicted score for input x, $\mathcal{D}$ a dataset, $\mathcal{D}_{attack}$ its attack subset, and $\mathcal{D}_{real}$ its real-access subset. The false acceptance rate FAR_τ over the dataset $\mathcal{D}$, for a given **acceptance threshold** τ is defined as:

$$FAR_\tau = \frac{1}{|\mathcal{D}_{attack}|} \sum_{x \in \mathcal{D}_{attack}} \mathbb{1}[f(x) < \tau], \tag{5.1}$$

where the function $\mathbb{1}[\cdot]$ is 1 if its argument is true, or 0 otherwise. In other words, false acceptance rate is the fraction of *attack* samples in the dataset that are wrongly classified as *real*. Unless otherwise specified, τ is assumed to be equal to 0.5.

False rejection rate (FRR)

The false rejection rate FRR_τ is analogously defined as:

$$FRR_\tau = \frac{1}{|\mathcal{D}_{real}|} \sum_{x \in \mathcal{D}_{real}} \mathbb{1}[f(x) \geq \tau]. \tag{5.2}$$

It is the fraction of *real-access* samples that are classified as *attack*.

Half total-error rate (HTER)

The half total-error rate $HTER_\tau$ is defined as:

$$HTER_\tau = \frac{FAR_\tau + FRR_\tau}{2}. \tag{5.3}$$

As such, it is the mean of the two types of errors for the same threshold, and is useful as a summary measure of the overall performance of the algorithm.

Equal error rate (EER)

FAR_τ, FRR_τ and $HTER_\tau$ always depend on an underlying threshold τ. As a single threshold-independent metric, the equal error rate (EER) is often used to compare models. For our purposes, we use the following definition:

$$EER = HTER_t \mid t \text{ satisfies } \min_t \left(FAR_t - FRR_t \right). \tag{5.4}$$

In other words, EER is the error at the threshold point t where $FAR_t = FRR_t$.

APCER, BPCER and ACER

Attack presentation classification error rate (APCER) and *bona-fide classification error rate* (BPCER) are recently proposed metrics, that were specifically thought of for the evaluation of presentation-attack detection methods. They are now part of ISO standard ISO/IEC 30107-3:2017 [33]. BPCER is analogous to FRR. $APCER_{PAIS}$, on the other hand, is similar to FAR, but is calculated for each *presentation-attack instrument species* (PAIS) separately. An aggregate version, $APCER_{AP}$ is defined as the maximum $APCER_{PAIS}$ among instruments at a given *attack potential* (AP), which considers how likely that attack is. In other words, its goal is to model a *worst-case scenario*.

The *average classification error rate* (ACER) is based on BPCER and APCER, and is used to assess the efficacy of algorithms in the official protocols of the OULU-NPU dataset [11]. In this context, BPCER is the same as FRR. APCER, on the other hand, is defined as the highest FAR when considering each attack variety in the dataset separately. ACER is then analogous to HTER, and is defined as the average of BPCER and APCER.

A note on frame scores versus video scores

Benchmarks created for evaluating face PAD algorithms differ in whether they count the whole video as a single example for evaluation purposes, or each frame separately. Unless otherwise noted, OULU-NPU evaluations are based on video scores, while RECOD-MPAD evaluations are based on frame scores. Because the proposed methods and baselines in this work always generate scores for static frames, video scores for the OULU-NPU protocols are computed as the average score over each predicted frame. This makes comparison with other methods possible and fair.

5.3 Baseline methods

In order to validate our proposal, we also define two baseline methods to be compared to ours, when considering RECOD-MPAD. The first one is a recently published handcrafted method [12] that combines both texture and color characterizations, and was shown to outperform other popular texture-based methods. The second baseline is a simplified version of Method I introduced in Chapter 4, in which only the classification layer is learned, with the pre-trained frozen core serving as a feature extractor.

5.3.1 Handcrafted baseline: color LBP

As a representative of handcrafted approaches, to be used as baseline, we choose a recently proposed color-texture method [12]. It is based on the local-binary pattern (LBP) texture descriptor [54], which is also at the core of many other PAD methods [51, 52, 24, 40, 59]. It differs from previous uses of LBP by considering multiple alternative color channels, instead of focusing on grayscale images or RGB channels. Because of that, we refer to it as the *color-LBP method* in this work.

Uniform LBP descriptor

At the core of the method is the basic $LBP_{P,R}$ operator, which transforms a single image channel of size $H \times W$ into a map of the same size, with codes corresponding to local texture patterns. The pattern at each location (x, y) is defined by the following equation:

$$LBP_{P,R}(x,y) = \sum_{i=0}^{P-1} \delta(r_i - r_c) \times 2^{i-1}, \tag{5.5}$$

where R is the radius of a circular neighborhood in the original image, centered at (x, y), from which P points are sampled. r_c refers to the pixel at (x, y), while r_i is i'th pixel in the neighborhood. The thresholding function $\delta(r_i - r_c) = 1$, if $r_i - r_c \geq 0$, or 0 otherwise.

These patterns can be compactly represented by P bits. For example, for $P = 8$, the pattern 00000000_2 corresponds to the case in which the center pixel is greater than all 8 pixels in its neighborhood.

In practice, a variation of the basic LBP called *uniform LBP* is used. In this case, the so-called *uniform* patterns are separated from the *non-uniform*. A pattern is uniform if its binary representation has at most two 0/1 transitions. For example, patterns 00000000_2, 00111000_2, 00111111_2 are uniform, but 01000010_2 and 00110100_2 are not.

The final LBP descriptor is a normalized histogram recording the frequencies of each pattern. For a uniform $LBP_{8,1}$ map, the histogram would consist of 58 bins for uniform patterns, and a single bin for all non-uniform patterns.

Frame pre-processing

For each frame, a square region containing only the centered user face is cropped and rescaled to 64×64, as in the original paper. The resulting RGB image is then converted to two different color spaces: HSV, with hue, saturation and value channels, and YCbCr, with luminance, chrominance blue, and chrominance red channels.

Feature vector

From each of the six HSV and YCbCr channels, a uniform $LBP_{8,1}$ histogram with 59 bins is computed. The final feature vector is defined as the concatenation of all six histograms, for a total of 354 dimensions.

Training an SVM classifier

Feature vectors computed from each frame in the training set are used to train a binary support-vector machine classifier (SVM) [9, 19] to separate genuine from attack frames. In our experiments, we use radial basis function kernels (RBF), which results in a non-linear classifier. Hyperparameters are tuned with a validation set.

The color-LBP method based on HSV and YCbCr achieves 6.2% EER on the overall protocol of the CASIA dataset, which is significantly better than similar characterizations that use grayscale images or RGB channels, achieving 24.8% and 16.1% EER, respectively.

It also performs better than chance on cross-dataset experiments considering REPLAY-ATTACK and CASIA, while most previous methods completely fail [12].

5.3.2 Pre-trained CNN as a feature extractor

In addition to the handcrafted baseline, we also evaluate the potential of using a pre-trained CNN baseline, in the spirit of transfer learning.

We use almost the same methodology described for Method I, with the same architecture, and with parameters up to layer *fire9* pre-trained on ImageNet. During training, instead of optimizing the whole network, we freeze the parameters of the core part of the architecture and optimize only the linear classification layer. We can interpret this as using the pre-trained network as a non-linear feature extractor that generates a more disentangled image representation, which is then used to train a linear classifier. Otherwise, data is pre-processed exactly as in Method I, and the training procedure is also the same, with the same hyperparameters.

This kind of transfer learning scheme was shown to work better than optimizing the whole network when the original and target problems are related and limited data is available [69]. It also trains much faster and requires much less memory, since only the gradient with respect to parameters in the last layers must be computed, and intermediate activations must not be saved.

Chapter 6

Experimental Results

In this chapter, we evaluate the efficacy of the proposed methods by performing careful experimentation with the RECOD-MPAD dataset. We also present results for the OULU-NPU dataset, which was used in a recent face PAD competition at the IJCB-2017 conference. In each section, we start by presenting the experimental protocol. We then proceed to describing any relevant details about the tested methods, if applicable. Finally, we present and discuss the results.

6.1 Overall results on RECOD-MPAD

In this section, we describe an experiment comparing the proposed methods and the baselines by using all available data on the RECOD-MPAD dataset. This protocol is similar to the official protocol in most other existing datasets, in the sense that it is user-disjoint, but otherwise includes all factors present in the data.

Protocol definition

In RECOD-MPAD's overall protocol, we include 64 frames from each video, as described in Section 5.1.1. These frames are pre-aligned face crops of maximum resolution, which ensures every algorithm starts from a common ground. In summary, there are 143,997 frames covering 45 users, 2 sensor devices, 5 sessions or illumination scenarios, and 4 attack types. These frames are divided into 3 user-disjoint subsets:

- A training set, containing a total of 76,798 frames from 24 users;
- A validation or development set, containing a total of 19,200 frames from 6 users;
- A test set, containing a total of 47,999 frames from 15 users.

The testing set is to be used exclusively for reporting the final results of a method. On the other hand, the validation set can be used freely for any kind of hyperparameter tuning, as an early stopping criterion, or simply for monitoring the training procedure. In general, the HTER, FAR, and FRR metrics at the standard threshold of 0.5 should be calculated on a per-frame basis, and reported for both subsets.

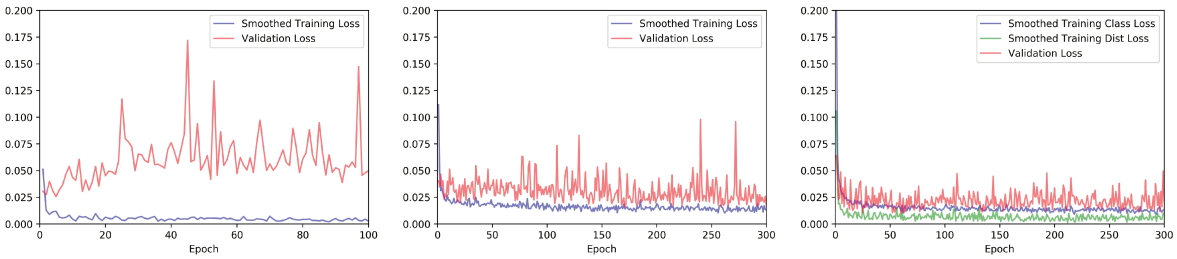
Tested methods

In this experiment, we evaluate the performance of both baselines, and of the three proposed methods.

Details of the Color LBP baseline are discussed in Section 5.3.1. Its purpose is to serve as a simple but effective representative of handcrafted approaches, since it has been shown to outperform most other methods in PAD benchmarks [12]. The validation set is used to tune hyperparameters of the SVM classifier through grid search, which are then used to fit a classifier to all available training data. The efficacy of the resulting classifier is then evaluated on the test set.

The networks of the three proposed methods and of the pre-trained baseline method were trained with all available frames in the training set. Validation data was used in an initial experimentation phase to find suitable hyperparameters, which were then fixed, as described in Section 4.1. Validation data was also used to monitor training curves, and the final model is selected as the one that achieved the smallest $HTER_{0.5}$ in the validation set, after a certain number of epochs. Methods II and III were trained for at most 300 epochs, while Method I and the pre-trained baseline were trained for at most 100 epochs, as there were no observable benefits in continuing training beyond that point.

Figure 6.1 shows the overall training curves of each method, in terms of training and validation loss as a function of epochs of training. In general, training looks much more stable for methods II and III.



(a) Method I - Whole-face CNN. (b) Method II - Patches CNN. (c) Method III - Spoof-loss CNN.

Figure 6.1: Training curves for the proposed methods, trained with RECOD-MPAD. Blue, and red curves depict the evolution of cross-entropy loss for training and validation data, respectively. For Method III, the green curve additionally represents the triplet loss component of the spoof loss.

Results

Table 6.1 presents the results for the described protocol. In general, all three proposed methods outperformed the baselines by a large margin. It is also noticeable how even a CNN whose core representations have been learned on another task can reach performance figures similar to those of the handcrafted baseline.

Table 6.1: Results for the overall protocol of RECOD-MPAD.

	Validation Set			Test Set		
Method	HTER	FAR	FRR	HTER	FAR	FRR
Color-LBP	3.87	1.02	6.72	4.94	0.72	9.16
Pre-trained CNN	3.62	2.62	4.61	6.59	2.32	10.86
Whole-face CNN (I)	0.91	0.40	1.43	0.82	0.35	1.29
Patches CNN (II)	**0.56**	0.67	0.44	1.14	0.34	1.94
Spoof-loss CNN (III)	**0.35**	0.29	0.42	**0.63**	0.12	1.15

Performance on the validation set suggests that, in comparison to Method I (Whole-face CNN), Methods II (Patches CNN) and III (Spoof-loss CNN) can more closely model the problem, in the sense that they reach points of lower validation error during training. This is a pattern that will also be evident in the next experiments. Although Method II reached lower validation error than Method I, the latter still performed better on the test set. On the other hand, compared to the other two methods, Method III performed much better, making less false-rejection errors, and roughly a third of false-acceptance errors.

6.2 Cross-factor experiments with RECOD-MPAD

The goal of the overall experiment described in Section 6.1 was to assess how well the tested methods could generalize to new users, and more data in general, but assuming similar acquisition and attack conditions. That protocol is most similar to how traditional PAD methods are evaluated. From the results, we could conclude that our proposed methods are better than the baselines. Equally important, we could validate their potential to be used as a data-driven tool for more challenging experimentation.

In the next experiments, we report only results for the three proposed methods. Each experiment defines a protocol in which we train and validate with only a subset of what we call the *factors* in the dataset, i.e., sessions, attack types, sensor devices, etc., leaving the other variables to make up the test dataset. As usual, the test set is to be used exclusively for reporting the final results of a method.

Models were trained by following the same procedure and hyperparameters described in Section 6.1 and Chapter 4. Because of that, each of the following subsections only describes protocols and results.

6.2.1 Cross-session protocol

In the cross-session experiment, our goal is to test how well models trained only in a limited number of illumination scenarios can generalize to new conditions. This is particularly important, since it is assumed that any useful method should work in all typical illumination scenarios. Unfortunately, this is one of the most overlooked aspects when constructing PAD datasets.

Protocol definition

We start from the same user-disjoint subsets defined in the overall protocol, with 24, 6, and 15 users in the training, validation and test sets, respectively. We then create a total of 5 cross-session sub-protocols by using a leave-one-session-out strategy. In other words, for session $i \in \{1, 2, 3, 4, 5\}$, sub-protocol i retains only session i in its test set, with frames from the remaining session making up the training and validation sets. This enables us to test how difficult it is to separate genuine from attack frames in each illumination scenario. More importantly, it can suggest how important it is to include examples from those scenarios in the training set. Details about each session are given in Section 5.1.1.

Results

By examining Table 6.6 we see the numbers for each sub-protocol are quite different. Most noticeable, the results for sub-protocol 5 suggest that models do not generalize so well to the extreme unseen conditions in session 5. False-rejection rate for Method I, for example, is 43.54%. This highlights the importance of accounting for these situations when developing PAD methods or building training and evaluation datasets.

From the point of view of model evaluation, we can see that there is no clear winner. For example, Methods I (*Whole-face CNN*) and II (*Patches CNN*) perform incredibly well in comparison to the others in sub-protocols 2 and 1, respectively. In fact, the numbers suggest that they perform better at the unseen test scenario than in the validation set, in each case. On the other hand, in sub-protocol 5, Method III (*Spoof-loss CNN*) is the only one that could be said to be acceptable, although false-rejection rate is still not low enough for industry standards.

Table 6.2: Results on RECOD-MPAD: protocol cross-session 1.

	Validation Set			Test Set		
Method	HTER	FAR	FRR	HTER	FAR	FRR
Whole-face CNN (I)	1.07	0.54	1.60	0.85	0.56	1.15
Patches CNN (II)	0.57	0.38	0.75	**0.34**	0.36	0.31
Spoof-loss CNN (III)	**0.34**	0.22	0.46	3.57	0.26	6.88

Table 6.3: Results on RECOD-MPAD: protocol cross-session 2.

	Validation Set			Test Set		
Method	HTER	FAR	FRR	HTER	FAR	FRR
Whole-face CNN (I)	1.53	0.92	2.15	**0.66**	0.34	0.99
Patches CNN (II)	0.47	0.71	0.23	2.66	0.31	5.00
Spoof-loss CNN (III)	**0.40**	0.48	0.33	1.77	0.00	3.54

Table 6.4: Results on RECOD-MPAD: protocol cross-session 3.

	Validation Set			Test Set		
Method	HTER	FAR	FRR	HTER	FAR	FRR
Whole-face CNN (I)	0.94	0.70	1.17	6.99	2.58	11.41
Patches CNN (II)	0.51	0.80	0.23	5.27	3.15	7.40
Spoof-loss CNN (III)	**0.28**	0.41	0.16	**4.54**	0.27	8.80

Table 6.5: Results on RECOD-MPAD: protocol cross-session 4.

	Validation Set			Test Set		
Method	HTER	FAR	FRR	HTER	FAR	FRR
Whole-face CNN (I)	1.09	1.07	1.11	1.73	1.99	1.46
Patches CNN (II)	0.45	0.54	0.36	**1.41**	1.21	1.61
Spoof-loss CNN (III)	**0.40**	0.51	0.29	2.85	0.27	5.42

Table 6.6: Results on RECOD-MPAD: protocol cross-session 5.

	Validation Set			Test Set		
Method	HTER	FAR	FRR	HTER	FAR	FRR
Whole-face CNN (I)	0.42	0.08	0.75	21.82	0.09	43.54
Patches CNN (II)	0.24	0.29	0.20	16.07	0.27	31.87
Spoof loss CNN (III)	**0.08**	0.06	0.10	**12.15**	0.55	23.75

6.2.2 Cross-attack protocol

In RECOD-MPAD's cross-attack protocol, the goal is to evaluate how well presentation-attack detection methods generalize to unseen attack variations. This is done by separating the available data into attack-disjoint subsets. More specifically, we build training and validation sets that only contain a subset of the attack varieties, while remaining attacks make up the test set.

Protocol definition

We start with the same strategy described in Section 6.1 for dividing frames into user-disjoint training, validation and test sets. Each subset contains only data from 24, 6 and 15 users, respectively. We then further filter these subsets to create 4 sub-protocols. In each sub-protocol, we select one type of display attack and one type of print attack to be

part of the training and validation sets, while the test set is left only with the other two remaining attacks. We call the display attacks *CCE42* and *HP17*, referring to the monitor brand and size, for short. The two varieties of print attacks are called *print1* and *print2*, and refer to the printed-photo attacks recaptured with more or less light, respectively. More details about the dataset can be found in Section 5.1.1. Table 6.7 summarizes how attacks are separated among subsets in each sub-protocol.

Table 6.7: Distribution of attack types in RECOD-MPAD's cross-attack sub-protocols.

	Training		Validation		Test	
Sub-protocol	Attacks	#frames	Attacks	#frames	Attacks	#frames
1	CCE42 print1	46,079	CCE42 print1	11,520	HP17 print2	28,799
2	HP17 print1	46,079	HP17 print1	11,520	CCE42 print2	28,799
3	CCE42 print2	46,079	CCE42 print2	11,520	HP17 print1	28,799
4	HP17 print2	46,079	HP17 print2	11,520	CCE42 print1	28,799

Results

Tables 6.8 to 6.11 summarize the results for the cross-attack protocol.

Table 6.8: Results on RECOD-MPAD: protocol cross-attack 1.

	Validation Set			Test Set		
Method	HTER	FAR	FRR	HTER	FAR	FRR
Whole-face CNN (I)	0.83	0.47	1.20	**3.47**	5.56	1.38
Patches CNN (II)	0.27	0.23	0.31	5.30	9.77	0.82
Spoof-loss CNN (III)	**0.23**	0.25	0.21	4.67	7.90	1.44

Table 6.9: Results on RECOD-MPAD: protocol cross-attack 2.

	Validation Set			Test Set		
Method	HTER	FAR	FRR	HTER	FAR	FRR
Whole-face CNN (I)	0.75	0.79	0.70	**11.52**	20.61	2.43
Patches CNN (II)	**0.24**	0.20	0.29	12.70	24.43	0.98
Spoof-loss CNN (III)	0.31	0.22	0.39	11.76	22.03	1.48

Table 6.10: Results on RECOD-MPAD: protocol cross-attack 3.

	Validation Set			Test Set		
Method	HTER	FAR	FRR	HTER	FAR	FRR
Whole-face CNN (I)	0.95	0.44	1.46	2.77	4.26	1.29
Patches CNN (II)	0.46	0.51	0.42	**1.81**	2.74	0.88
Spoof-loss CNN (III)	**0.31**	0.43	0.18	3.37	5.49	1.25

Table 6.11: Results on RECOD-MPAD: protocol cross-attack 4.

	Validation Set			Test Set		
Method	HTER	FAR	FRR	HTER	FAR	FRR
Whole-face CNN (I)	1.02	0.59	1.46	12.64	23.09	2.19
Patches CNN (II)	0.59	0.65	0.52	11.97	22.72	1.22
Spoof-loss CNN (III)	**0.32**	0.46	0.18	**10.43**	19.54	1.32

In general, models trained with attacks from the larger display demonstrated acceptable generalization when predicting unseen attacks performed by the smaller monitor. The opposite did not happen. In fact, models trained with the 17-inch monitor are only slightly better than chance at detecting attacks from the larger display. They account for most of the false-acceptance errors in sub-protocols 2 and 4.

Attacks performed with a smaller monitor and recaptured with a fixed-focus sensor tend to have limited resolution, due to soft focus at closer distances. There are certainly other factors that could account for the difference between the two display attacks, but we hypothesize that when training only with lower resolution attacks, models are more likely to get biased to the difference in resolution, instead of other differences between real-access and attack images. This highlights the importance of having more realistic attacks in the training set.

6.2.3 Cross-device protocol

In this experiment, our goal is to test how well models trained only with a subset of the available sensor devices generalize to the difficult task of distinguishing between real-access and attack frames from unseen devices.

Protocol definition

We start by separating users into training, validation and test sets, exactly as in the overall protocol. Since RECOD-MPAD has two sensor devices, we create two sub-protocols, each with roughly half of the available data. Sub-protocol 1 has only frames from device 2 (*Moto*

X Style) in the training and validation sets, and only frames from device 1 (*Moto G5*) in the test set. Sub-protocol 2 has only frames from device 1 (*Moto G5*) in the training and validation sets, and only frames from device 2 (*Moto X Style*) in the test set.

Results

Tables 6.12 and 6.13 summarize the results for this experiment. The immediate observation is that models trained only with device 2 (*Moto X Style*) generalize relatively better than the other way around, when faced with the problem of distinguishing real-access attempts from attacks of device 1 (*Moto G5*).

Table 6.12: Results on RECOD-MPAD: protocol cross-device 1.

	Validation Set			Test Set		
Model	HTER	FAR	FRR	HTER	FAR	FRR
Whole-face CNN (I)	0.24	0.27	0.21	8.58	8.18	8.98
Patches CNN (II)	0.10	0.10	0.10	6.66	7.28	6.04
Spoof-loss CNN (III)	**0.06**	0.07	0.05	**6.33**	4.00	8.67

Table 6.13: Results on RECOD-MPAD: protocol cross-device 2.

	Validation Set			Test Set		
Model	HTER	FAR	FRR	HTER	FAR	FRR
Whole-face CNN (I)	0.55	0.53	0.57	16.57	27.37	5.77
Patches CNN (II)	**0.19**	0.22	0.16	17.98	18.29	17.67
Spoof-loss CNN (III)	**0.19**	0.22	0.16	**8.84**	8.49	9.19

To find explanations, we could start by looking at the differences between the two cameras. Similarly to the situation with the cross-attack protocol, the most obvious difference is in resolution. Device 1, used for training in sub-protocol 2, has a lower capacity of resolving finer details at the considered distances. This, again, could encourage models to learn a more artificial separation between real and attack frames, that does not generalize to new situations in which attacks are more similar in quality to real frames. That quality depends on the sensor camera itself interacting with attack variables. Ironically, a camera with focus fixed at longer distances would be bad at capturing good-quality close-up pictures, but this could aid in detecting presentation attacks performed with smaller display or paper sizes.

In terms of individual model performance, we note that in sub-protocol 1 the two methods based on training with patches performed better than the whole-face method. Moreover, the boundaries at the standard threshold of 0.5 seem to be complementary,

in the sense that one model makes less false-rejection errors, while the other makes less false-acceptance errors.

In sub-protocol 2, although Methods I and II undoubtedly perform much worse than in sub-protocol 1, Method III performs surprisingly better than the other two. The multi-objective loss and the associated training procedure were envisioned chiefly as a naturally more sensible way of training with multiple sensor devices, but this experiment suggests that there is a benefit even when training with a single device. This is probably due to the way the triplet loss component encourages a similarity between real samples and an explicit dissimilarity between real and attack samples in the learned space, which is different than just separating two classes.

6.2.4 Controlled protocol

Finally, we also present a special experimental protocol in which we would like to evaluate what happens when we train only with "well-behaved" frames, excluding most extremely distorted printed-photo frames and all frames in session 5, which are much darker, noisier and blurrier than other frames.

Protocol definition

This protocol uses the same disjoint subsets of users as in the overall experiment in Section 6.1 to separate data into training, validation and test subsets. The test set remains exactly the same, which makes it possible to directly compare with the numbers reported for that protocol.

The training and validation sets, however, are filtered so that all frames from session 5 are removed. In addition to that, the first 32 frames from all printed-attack videos, corresponding to the first half of these recordings, are also removed. As described in Section 5.1.1, printed-photo videos start with the attacker deliberately distorting or warping the paper while moving it away from the camera, which makes the cropped face area very small. The eliminated frames correspond mostly of the frames in those parts of the video.

Results

Results for this experiment are shown in Table 6.14. Although the *whole-face CNN* performed relatively bad in this scenario, the other methods, based on training with patches, showed promising generalization potential.

In the aggregate, we see that there is a deterioration in performance, as compared to test error in the overall protocol (Section 6.1). This suggests that training with distorted and low-quality frames does not hurt overall performance, while removing them from training can make models fail when encountering the situations they represent in the real world.

Table 6.14: Results on RECOD-MPAD: protocol controlled.

	Validation Set			Test Set		
Model	HTER	FAR	FRR	HTER	FAR	FRR
Whole-face CNN (I)	0.40	0.14	0.65	4.76	0.44	9.07
Patches CNN (II)	0.28	0.24	0.33	**1.86**	0.38	3.33
Spoof-loss CNN (III)	**0.17**	0.25	0.10	2.20	0.58	3.81

6.3 Experiments with OULU-NPU

Veering away from the experiments using RECOD-MPAD, we now focus on an analysis considering the OULU-NPU dataset. Some important details about the dataset were already given in Sections 2.2 and 5.1.2. We also refer the reader to the original paper in which it was proposed [11].

In the following subsections, we describe each of its official user and factor-disjoint protocols, with the accompanying results and discussion. For that, we draw on the results of the IJCB-2017 competition [10], briefly mentioned in Section 5.1.2. During the analysis, it would be unfeasible to list the results of all participants in the competition. Therefore we decided to add only the performance figures of the top-3 entries in each protocol, considering one entry per team. In addition to that, we also include the official baseline and our own entry in the competition.

The competition baseline is based on the same method we described in Section 5.3.1, but with small changes in face alignment, and a different type of classifier at the end: a Softmax classifier, instead of an SVM with RBF kernel. In practice, we compared the accuracy of our own implementation to the competition implementation and they were very similar.

Our entry in the competition is considerably different than the solutions we propose in this dissertation. At that time, we also used SqueezeNet as our core architecture, but there are several differences in the overall pipeline. Most importantly, our competition entry was based on the mean score of the LBP baseline and the CNN classifier. The CNN classifier itself was based on pre-training with the aligned whole-face crops from the UVAD and CASIA datasets, selecting the checkpoints with the lowest validation error, and then retraining with the appropriate OULU-NPU subset.

In addition to our three proposed methods, in this section we also include the results for combining the predictions of Methods II and III, and all three methods. The late fusion strategy consists in computing the arithmetic average of the final scores. The idea is to give a glimpse of the complementarity of these methods, and to estimate how much we could gain in accuracy by running the multiple networks in an operational scenario.

Evaluation metrics

On each protocol, evaluation is based on the equal error rate (EER) and the attack classification error rate (ACER), as described in Section 5.2. More specifically, EER should be computed and reported on the validation set (officially called development set), which happens at a specific score threshold t. Subsequently, this threshold is used to compute the $ACER_t$ on the test set, which is the main metric used to compare different solutions. All metrics are calculated considering each video as a single example, instead of frames. For our solutions, video scores are calculated as the average score of sampled frames in that video, which makes for a fairer comparison.

6.3.1 Cross-session protocol

In the cross-session protocol of OULU-NPU (Protocol I), the goal is to evaluate generalization to unseen illumination conditions.

Protocol definition

Since the full dataset considers videos from 3 sessions, in this protocol training and validation sets are filtered to contain only videos from the first two sessions, while the test set is then composed of videos from the remaining session. Unfortunately, the official protocol does not test generalization to the other two available illumination conditions. Training, validation, and test sets are user-disjoint, with 20, 15, and 20 users, respectively. In total, there are 1200, 900, and 600 videos in the training, validation and test sets, respectively.

Results

Table 6.15 summarizes the results. Compared to our other solutions, we see that Method II (*Patches CNN*) is the best performer in this case, being significantly better than the competition baseline. In general, it is among the top-performing methods in the context of the competition, and would place us in position 3 out of 13 participants. In fact, our actual entry was already the third best. If we consider score-level fusion, combining Methods II and III would put us at the second place, while combining all three methods would surpass all participants in the competition.

Although all three sessions in the dataset are similar, in the sense that they are controlled static indoor sessions, this protocol still proved to be a challenge, which highlights the fragility of PAD methods in general.

Table 6.15: Results for protocol I of OULU-NPU (cross-session).

Method	Validation EER (%)	Test ACER (%)
Whole-face CNN (I)	2.6	8.3
Patches CNN (II)	**0.3**	**7.5**
Spoof-loss CNN (III)	0.8	8.8
Score fusion (II + III)	0.8	6.7
Score fusion (I + II + III)	0.8	**6.2**
Competition baseline	4.4	12.9
Competition 1st	0.7	6.5
Competition 2nd	0.6	6.9
Competition 3rd	2.2	8.3
Our entry (different method)	2.2	8.3

6.3.2 Cross-attack protocol

In the cross-attack protocol (Protocol II) of OULU-NPU, the goal is to test generalization to new attack conditions.

Protocol definition

Attack videos created by using display 1 and printer 1 are selected to compose the training and validation subsets, while the test set is made up only from attack videos that use display 2 and printer 2. Users are distributed among subsets exactly as in Protocol I.

Results

Table 6.16 shows results for this protocol. We note that our two methods trained with patches (*Patches CNN* and *Spoof-loss CNN*) show significantly lower error rates than the *Whole-face CNN* method. With an ACER of 7.2%, they lead to less than half the errors the baseline would make, and are slightly worse than the third place in the competition. A simple score-level fusion of Methods II and III would be placed only 0.1% behind the second best in the competition.

Table 6.16: Results for protocol II of OULU-NPU (cross-attack).

Method	Validation EER (%)	Test ACER (%)
Whole-face CNN (I)	3.7	11.3
Patches CNN (II)	**0.3**	**7.2**
Spoof-loss CNN (III)	0.7	**7.2**
Score fusion (II + III)	**0.4**	**6.2**
Score fusion (I + II + III)	1.1	7.6
Competition baseline	4.1	14.6
Competition 1st	0.9	2.5
Competition 2nd	1.3	6.1
Competition 3rd	4.4	6.7
Our entry (different method)	3.7	10.0

6.3.3 Cross-device protocol

The goal of the cross-device protocol of OULU-NPU (Protocol III) is to test generalization across sensor devices.

Protocol definition

As usual, subsets are user-disjoint, and users are distributed exactly as in the other protocols. In this case, however, a leave-one-device-out strategy is used to create 6 sub-protocols. In each sub-protocol, a single sensor device is left out for testing, while videos from the remaining 5 devices are used to compose the training and validation sets. In each of the 6 sub-protocols, there are 1500, 1125, and 300 videos making up the training, validation, and test sets, respectively.

Instead of reporting EER and ACER individually, the mean and standard deviation of these metrics over each sub-protocol are to be reported.

Results

Table 6.17 presents the results for this protocol. We notice two patterns. Firstly, this is the first protocol in which the proposed solutions are behind the baseline. Most importantly, on average we see that Method II (*Patches CNN*) performed better than Method I (*Whole-face CNN*), and Method III (*Spoof-loss CNN*) performed better than the other two. Fusion did not improve performance, suggesting that Method III was strictly better in this case. Although variance is high, this still offers some support to the hypothesis that Method III can better deal with multiple devices during training, and that patches-based methods can model more generalizable clues.

This dataset was constructed with cameras that are very different from one another. Arguably, generalization across very different devices is the less important goal. After all, we do expect a useful model to work reasonably well under typical, but unseen lighting conditions, or even in face of a new attack instrument. But it is not surprising that a

trained model can fail if deployed on a system in which the sensor is very different than the one it was trained with, given the low-level nature of the problem.

Table 6.17: Results for protocol III of OULU-NPU (cross-device).

Method	Validation EER $\pm\sigma$ (%)	Test ACER $\pm\sigma$ (%)
Whole-face CNN (I)	4.5 ± 0.9	19.7 ± 5.8
Patches CNN (II)	$\mathbf{0.6 \pm 0.3}$	15.6 ± 6.9
Spoof-loss CNN (III)	$\mathbf{0.6 \pm 0.3}$	$\mathbf{13.6 \pm 7.0}$
Score fusion (II + III)	$\mathbf{0.5 \pm 0.3}$	$\mathbf{13.6 \pm 6.7}$
Score fusion (I + II + III)	0.6 ± 0.5	14.2 ± 6.5
Competition baseline	3.9 ± 0.7	11.4 ± 4.6
Competition 1st	0.9 ± 0.4	$\boxed{3.8 \pm 2.4}$
Competition 2nd	1.4 ± 0.5	6.5 ± 4.6
Competition 3rd	0.9 ± 0.4	7.4 ± 3.3
Our entry (different method)	2.9 ± 0.7	9.6 ± 6.7

6.3.4 Cross-* protocol

In Protocol IV of OULU-NPU, all factors are considered. Because of that, we call it the cross-* protocol. Its goal is to evaluate how well models generalize to totally unseen conditions: new sensor device, new illumination, and new attack instrument. As such, it is the hardest protcol.

Protocol definition

Protocol definition is analogous to Protocol III, with each test set containing only one of the sensor devices. Moreover, test sets contain only videos taken in session 3, while training and validation sets contain only videos from sessions 1 and 2. Attacks are also divided, with printed photo 1 and display attack 1 making up the training and validation sets, and printed attack 2 and display attack 2 left for the test set. In each sub-protocol, training, validation, and test sets have 600, 450, and 60 videos, respectively.

Results

The overall results for Protocol IV are shown in Table 6.18. As expected, all models show very high ACER in the test set, with high variance. Nonetheless, all three of our proposed solutions compare favorably to methods presented in the competition, especially Method III (*Spoof-loss CNN*). If we were to submit any of our solutions to the competition, we would take the second place out of 13. In particular, mean error and standard deviation are much lower than those of other methods based on feature learning.

Table 6.18: Results for protocol IV of OULU-NPU (cross-*).

Method	Validation EER $\pm\sigma$ (%)	Test ACER $\pm\sigma$ (%)
Whole-face CNN (I)	3.9 ± 0.3	19.2 ± 10.5
Patches CNN (II)	$\mathbf{0.5 \pm 0.3}$	18.3 ± 7.5
Spoof-loss CNN (III)	$\mathbf{0.5 \pm 0.3}$	$\mathbf{14.2 \pm 6.1}$
Score fusion (II + III)	$\mathbf{0.4 \pm 0.2}$	$\mathbf{13.3 \pm 5.1}$
Score fusion (I + II + III)	0.6 ± 0.5	14.2 ± 6.5
Competition baseline	4.7 ± 0.6	26.3 ± 16.9
Competition 1st	1.1 ± 0.3	$\boxed{10.0 \pm 5.0}$
Competition 2nd	1.0 ± 0.4	22.1 ± 17.6
Competition 3rd	2.2 ± 1.7	22.1 ± 20.8
Our entry (different method)	3.7 ± 0.7	22.5 ± 18.2

6.3.5 Final remarks

With these experiments, we were able to understand how the proposed methods compare to the state of the art in a different but standardized PAD setting. The results are promising, especially for Methods II and III, which were better than Method I in all protocols.

Methods II and III compare favorably to methods that were part of the competition. The first place achieved significantly lower error rates than other entries, but is based on a handcrafted approach that uses temporal information. Some of the other methods also explored the temporal dynamics or background. We highlight that it is not clear whether these methods would work well under our constraints, in which a decision must be made from at most a few frames, and the user is not necessarily standing still. Considering that our approach is purely data-driven, frame-based, and not specifically tuned to this dataset in particular, the results are satisfying. More importantly, our proposed solutions showed better generalization than all except the first-place in the hardest protocol.

6.4 On-device user-specific adaptation

In this experiment, the goal is to evaluate how well the method for threshold adaptation described in Section 4.4 actually works in practice.

Protocol definition

We focus on two cases: the cross-attack scenario described in Section 6.2.2 and the cross-device scenario described in Section 6.2.3. We start from the *Spoof-loss CNN* models trained in cross-attack subprotocols 1 and 2, and cross-device subprotocols 1 and 2. For each of these 4 models, we partition the test set so that, for each user, approximately 128 real-access frames from 2 sessions play the role of the gallery, while the real and attack frames from the remaining 3 sessions are kept in the test set for error estimation. We then

report the total HTER on the new test set for the default threshold of 0.5, and the total HTER using the learned threshold. For convenience, although each user uses a different threshold, we report only the aggregate error of all 15 users in the test set. Threshold estimation is done as described in Section 4.4, with $\epsilon = 0.05$ and $\Delta = 0.05$.

Results

Results corresponding to the models trained in RECOD-MPAD's cross-attack subprotocols 1 and 2 are shown in Tables 6.19 and 6.20, respectively. As we can see, even by using only two sessions as gallery, overall HTER is reduced by as much as 33.4%.

Table 6.19: User-specific adaptation in a cross-attack scenario (1).

Gallery	Standard threshold HTER	User-specific threshold HTER	Error reduction (%)
1, 2	5.33	3.55	33.4
1, 3	4.19	2.89	31.0
1, 4	4.96	3.80	23.4
1, 5	4.52	3.62	19.9
2, 3	4.64	3.09	33.4
2, 4	5.40	4.06	24.8
2, 5	4.97	3.66	26.4
3, 4	4.26	3.73	12.4
3, 5	3.82	2.96	22.5
4, 5	4.59	4.05	11.8

Table 6.20: User-specific adaptation in a cross-attack scenario (2).

Gallery	Standard threshold HTER	User-specific threshold HTER	Error reduction (%)
1, 2	11.65	9.37	19.6
1, 3	9.35	8.16	12.7
1, 4	11.35	9.24	18.6
1, 5	14.37	11.25	21.7
2, 3	9.44	7.86	16.7
2, 4	11.45	9.35	18.3
2, 5	14.47	11.36	21.5
3, 4	9.15	7.98	12.8
3, 5	12.17	10.76	11.6
4, 5	14.17	11.25	20.6

Tables 6.21 and 6.22 show the results when considering the cross-device evaluations. In contrast to the previous case, here the benefit is not so evident or consistent. For the

model trained with the *Moto X* smartphone and tested with *Moto G5*, we see considerable gains when using session 1 as part of the gallery, but in other cases, error either increased or did not change so much. For the model trained with *Moto G5* and tested with *Moto X*, the situation is worse, with errors consistently increasing when trying to learn user-specific thresholds from only 2 sessions.

Table 6.21: User-specific adaptation in a cross-device scenario (1).

Gallery	Standard threshold HTER	User-specific threshold HTER	Error reduction (%)
1, 2	6.11	5.13	16.0
1, 3	6.10	4.65	23.8
1, 4	6.61	4.56	31.0
1, 5	6.11	5.95	2.6
2, 3	6.15	5.63	8.5
2, 4	6.66	6.28	5.7
2, 5	6.16	6.32	-2.6
3, 4	6.65	6.87	-3.3
3, 5	6.15	6.13	0.3
4, 5	6.66	6.56	1.5

Table 6.22: User-specific adaptation in a cross-device scenario (2).

Gallery	Standard threshold HTER	User-specific threshold HTER	Error reduction (%)
1, 2	11.25	12.28	-9.2
1, 3	9.32	11.41	-22.4
1, 4	10.56	11.58	-9.7
1, 5	6.40	6.61	-3.3
2, 3	10.29	13.58	-32.0
2, 4	11.53	13.55	-17.5
2, 5	7.36	7.29	1.0
3, 4	9.60	12.41	-29.3
3, 5	5.43	5.84	-7.6
4, 5	6.67	7.63	-14.4

Overall, the results suggest that learning user-specific cutoffs on the target device is beneficial, as long as the sensor camera is not too different from the ones seem during training. In fact, this chaotic behavior with cross-device protocols was already observed in previous experiments. Here we highlight the surprising reduction of error rates even when considering new attacks and a gallery consisting only of two sessions.

6.5 Examples of success and error cases

In this section, we present some success and error cases when training with RECOD-MPAD under the protocol described in Section 6.1. To enhance interpretability, we describe a procedure to generate heatmaps from class-specific activations in the final layers of a CNN, which is similar to the method used by Zhou et al. [82].

Heatmap generation

Fully-convolutional architectures avoid the traditional stack of fully-connected layers for classification at the end of deep neural networks. This greatly reduces the number of parameters, as already explained in Chapter 3. Because the final class-likelihood vector is generated by directly averaging class-specific activation maps, these architectures have another advantage: we can easily generate class-specific heatmaps, which highlight which regions in the image were most responsible for the prediction.

For Methods I and II, we consider the $2 \times 14 \times 14$ activation maps preceding the final global-average pooling (GAP) operation. At each spatial location, we generate normalized *local attack probabilities* by applying a modified Softmax function:

$$p\left(i,j\right) = \frac{e^{f_1(i,j)/T}}{e^{f_0(i,j)/T} + e^{f_1(i,j)/T}}, \tag{6.1}$$

where $f_1\left(i,j\right)$ represents the activation for the attack map at spatial position (i,j). This mimics the Softmax function that is applied to spatially averaged activations, when training the network with the cross-entropy loss (see Section 4.1). T is a so called "temperature" constant, which effectively flattens the resulting distribution for values greater than 1. We use $T = 10$ to make visualizations more nuanced. When the resulting heatmap image is rescaled and resized for visualization purposes, the differences in brightness highlight which regions in the original image were more important for classifying it as attack or genuine.

For Method III, the procedure is slightly different, since the spatial dimensions were already "eliminated" by GAP after the *fire9* layer, where maps still correspond to feature activations. Instead of considering class-specific activation maps directly, we therefore consider the 512 feature maps of size 14×14 from the *fire9* layer. Given the 512 attack weights in the classification layer, we generate an attack map by first multiplying each of the 512 *fire9* maps by the respective attack weight, and then summing over the feature axis. The map for the real-access class is generated analogously. The procedure for generating heatmaps is then exactly the same as before.

Visualizations

Here we present examples of success and error cases, with visualization of associated heatmaps. For each example, we show its min-max normalized heatmap, which helps to emphasize the relative differences in activation strength of different areas in the image. We also present a resized and overlaid heatmap, in which smoothed local probabilities are not scaled. In this case, the red channel of the overlaid heatmap is defined as $p(i,j)$,

according to Equation 6.1, the green channel as $1.0 - p(i, j)$, and the blue channel is 0. When local attack probabilities are high, that region is visualized as red, while very low attack probabilities are shown as green.

The following examples do not necessarily make for good comparisons between methods. With that in mind, they can certainly give valuable hints as to which facial, illumination or attack characteristics are more likely to help or deceive the networks in making their decision.

Figure 6.2 shows some examples of correct predictions for genuine input images, considering each method. One notices that most regions in the images contribute similarly to the correct prediction, but there are some patterns. For example, in Figure 6.2a, the eye region seems to be more discriminative, while the mouth region is more uncertain. Still the overall attack probability is 0%. In Figures 6.2b and 6.2f, the red areas indicate some confusion for certain combinations of lighting and glasses, but, again, this was not enough to cause a wrong prediction. Although these are only a few examples, they highlight the capability of the models to work in different lighting conditions.

(a) Method I: attack probability = 0.0000

(b) Method I: attack probability = 0.0000

(c) Method II: attack probability = 0.0000

(d) Method II: attack probability = 0.0001

(e) Method III: attack probability = 0.0003

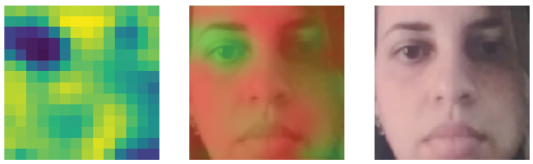
(f) Method III: attack probability = 0.0004

Figure 6.2: **True acceptance** examples, with heatmaps. Examples are taken from the validation set evaluations of the *overall protocol* in Section 6.1. Left: yellow/bright and blue/dark stand for higher and lower attack likelihood, respectively. Right: input image. Middle: overlaid heatmap.

Figure 6.3 shows examples of correctly predicted attack frames. As in the previous cases, we note that the final prediction is typically very certain. In Figure 6.3a, we observe much stronger activations in the eye region, possibly due to the glasses. In Figure 6.3d, the

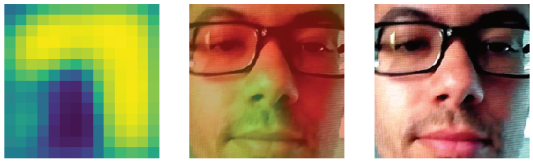
model was more sensitive to the dimmer side of the image. In the example in Figure 6.3b, the heatmaps suggest that the model perceived the left-eye region as indicative of real-access. Figures 6.3e and 6.3f suggest that eyebrows could also be confusing.

(a) Method I: attack probability = 1.0000



(b) Method I: attack probability = 0.9995

(c) Method II: attack probability = 1.0000



(d) Method II: attack probability = 0.9999

(e) Method III: attack probability = 1.0000

(f) Method III: attack probability = 1.0000

Figure 6.3: **True rejection** examples, with heatmaps. Examples are taken from the validation set evaluations of the *overall protocol* in Section 6.1. Left: yellow/bright and blue/dark stand for higher and lower attack likelihood, respectively. Right: input image. Middle: overlaid heatmap.

Figure 6.4 depicts some of the false-acceptance errors among validation frames, for each proposed method. Figures 6.4a and 6.4c suggest that the networks are sensitive to some kinds of reflections as attack clues, but these examples are still classified as genuine, mostly due to the regions around one of the eyes or eyebrows. In Figure 6.4b, the wrong prediction is mostly due to the regions around the eyes, while nose and hair areas were more revealing of the true label, but not as strongly. In Figure 6.4d, the dark, but content-wise rich region around eyes and nose were responsible for the wrong classification, while part of the mouth, and some of the flatter regions were more conducive to the true label.

In Figures 6.4e and 6.4f, we call attention to the fact that the regions around the eye on the reader's left-hand side were almost enough to make the CNN predict the label *attack*, but the contributions of the rest of the image "voted" for *genuine*. It is not clear why the network was much more sensitive to the eye on the left, in Figure 6.4e. On the other hand, the hair strand and the shadow caused by it could have contributed to the difference in Figure 6.4f. Noticeable, these are relatively high quality attacks.

(a) Method I: attack probability = 0.2294

(b) Method I: attack probability = 0.0327

(c) Method II: attack probability = 0.0532

(d) Method II: attack probability = 0.0200

(e) Method III: attack probability = 0.2641

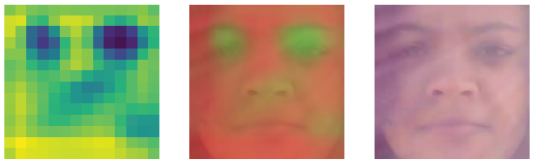
(f) Method III: attack probability = 0.0932

Figure 6.4: **False acceptance** examples, with heatmaps. Examples are taken from the validation set evaluations of the *overall protocol* in Section 6.1. Left: yellow/bright and blue/dark stand for higher and lower attack likelihood, respectively. Right: input image. Middle: overlaid heatmap.

Figure 6.5 illustrates some of the false-rejection errors. In Figures 6.5a, 6.5b, 6.4c, and 6.5e, we see that false classification as *attack* is mostly due to the effect of lens flare when shooting outdoors with direct sunlight. As pointed out in Section 5.1.1, the strong highlights and loss of contrast can make the resulting images look like presentation attacks. In Figure 6.5a, the region that most contributed to the false rejection was precisely the area around the sun-caused highlight, while the opposite higher-contrast region with facial hair still suggested that the frame was genuine. In Figures 6.4c and 6.4e, we also notice that the regions closer to the strong highlight had the highest activations for the *attack* label. In this case, Method III was somewhat more robust to the effect.

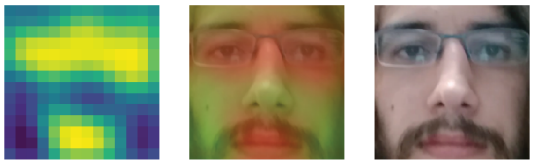
In Figure 6.5d, which depicts an indoor scenario, we see that the glasses or simply the eye region contributed significantly to the wrong prediction. Perhaps more puzzling, the mouth region was also seen as a strong indicator for rejection. In Figure 6.5f, it is clear that the reflection of the glass lens is what caused the wrong rejection.

(a) Method I: attack probability = 0.9117



(b) Method I: attack probability = 0.9998

(c) Method II: attack probability = 0.9518



(d) Method II: attack probability = 0.9801

(e) Method III: attack probability = 0.8818

(f) Method III: attack probability = 0.9990

Figure 6.5: **False rejection** examples, with heatmaps. Examples are taken from the validation set evaluations of the *overall protocol* in Section 6.1. Left: yellow/bright and blue/dark stand for higher and lower attack likelihood, respectively. Right: input image. Middle: overlaid heatmap.

We conclude this section by remembering that these are only a few representatives of success and error cases, and as such, they should not be used alone as a means of evaluating the quality of the models. Some mistakes hardly make sense, which could suggest problems. On the other hand, some of the errors were expected. In particular, we see that a significant portion of errors arise when operating in extreme but not unusual lighting conditions. This highlights the importance of considering realistic datasets when training and evaluating PAD models.

6.6 Extra: cross-dataset experiments

In Section 2.4, we highlighted the fact that the community is moving away from purely intra-dataset protocols as a means of evaluating PAD methods. In this work, we focused mainly on what we called cross-factor protocols, in which we partition the factors in a single dataset to get a more realistic view of failure situations. A much more challenging setting is training on one dataset and evaluating the model on another dataset, with completely different conditions. This typically results in an assessment that is overly pessimistic, but potentially useful.

In this section, we consider using OULU-NPU to evaluate models trained on RECOD-

MPAD, and, conversely, using RECOD-MPAD to test models trained on OULU-NPU.

Training on RECOD-MPAD and testing on OULU-NPU

We start by taking the models trained with all available training data from RECOD-MPAD. Details were discussed in Section 6.1. For the test set, we consider all videos from the test subset of OULU-NPU, i.e., all videos corresponding to users 36-55. To get more comparable and stable results, we compute all metrics based on the EER threshold of the whole set.

Results are shown in Table 6.23. We see that the numbers are generally not good, but this was expected. Still, performance is well above chance, except when presented with the second kind of print-attack in the dataset. This print attack is the sharper one between the two, which might explain the difference. In terms of predicting display attacks, the numbers are much better, which suggests that the algorithms were able to learn generalizable clues from the attacks present in RECOD-MPAD. There is however a difference between the two display attacks, with the first one proving to be harder to detect.

Table 6.23: Cross-dataset evaluation on OULU-NPU.

	BPCER	APCER			
Model	Real	Print 1	Print 2	Display 1	Display 2
Whole-face CNN (I)	27.22	36.67	52.78	11.11	8.33
Patches CNN (II)	29.17	35.83	52.22	18.89	9.44
Spoof-loss CNN (III)	25.28	28.89	48.89	16.39	6.67

Training on OULU-NPU and testing on RECOD-MPAD

Now we consider models trained on OULU-NPU. More specifically, we consider models trained for protocol I, described in Section 6.3. For the evaluation dataset we use all frames from the test partition of RECOD-MPAD, exactly as in Section 6.1. Here, however, we use the ERR threshold to calculate the other metrics.

The overall results are shown in Table 6.24. In general, we see that display attacks simulated with the 42-inch *CCE TV* (display 1) were much harder to detect than attacks with the smaller monitor, which is consistent with previous observations in the RECOD-MPAD cross-attack protocol. Surprisingly, the model corresponding to Method II was much better than Method III at detecting print attacks in this case.

Table 6.24: Cross-dataset evaluation on RECOD-MPAD.

	BPCER	APCER			
Model	Real	Print 1	Print 2	Display 1	Display 2
Whole-face CNN (I)	30.58	43.07	38.14	36.85	4.24
Patches CNN (II)	15.15	6.79	3.88	37.51	12.38
Spoof-loss CNN (III)	21.05	33.90	23.31	21.02	5.98

Table 6.25 shows the errors on the RECOD-MPAD dataset broken down by sessions. We see that although errors are mostly consistent between sessions for the *Whole-face CNN* model, there are some patterns. In general, frames from session 3 were easier to classify than frames from sessions 1, 2, and 4. This is consistent with the fact that the illumination conditions in session 3 are more similar to the indoor sessions in OULU-NPU. It is, however, surprising that Methods II and III were relatively successful at classifying frames from session 5, whose dimmer and noisier conditions are not well represented in OULU-NPU.

Table 6.25: Cross-dataset evaluation on RECOD-MPAD - errors by session.

	HTER				
Model	Session 1	Session 2	Session 3	Session 4	Session 5
Whole-face CNN (I)	34.53	30.59	28.25	29.77	29.75
Patches CNN (II)	19.97	15.08	13.63	19.15	7.90
Spoof-loss CNN (III)	25.55	21.86	16.47	28.54	12.86

6.7 Mobile implementation

From the start, one of our goals was to develop a method that not only could successfully detect attacks against face authentication in mobile devices, but could also be efficiently embedded in these devices. Therefore, in the context of our research project, we are also developing a working prototype that combines face authentication with presentation-attack detection. Here we provide evidence that our mobile implementation is feasible.

The CNN-part of our Android implementation is based on *TensorFlow* [1]. Assuming we forward a single image into the network, all methods have essentially the same running time, since the bulk of the computation is in the shared architectural core. Table 6.26 summarizes running time and peak memory usage for pre-processing and forwarding a single 227×227 RGB image into the network. This does not include face detection and alignment, but that is much faster and negligible in comparison to a forward pass through the network. Mean running time and standard deviation is calculated over 20 independent

runs. We highlight that the following figures are for non-optimized code, so improvement is to be expected.

Table 6.26: Computational demands of the mobile implementation.

Device	Release date	CPU	PAD time (ms)	Peak memory
Moto Z Play	09/2016	8-core 2.0 GHz	197.75 ± 34.57	<50MB
Moto G5S Plus	08/2017	8-core 2.0 GHz	233.85 ± 28.55	<50MB

From these numbers, we observe that the proposed methods can run in modern smartphone devices in under one second, which would be almost instantaneous and thus transparent to most users. In fact, it is still feasible to run two or forward steps in sequence, which would be interesting in a score fusion scheme, exploring the complementarity of the trained models. The small amount of required memory ensures that the authentication step does not interfere with other applications.

Chapter 7

Conclusion and Future Work

In this work, we proposed three different ways of training a convolutional neural network to model and solve the face presentation-attack detection problem in a completely data-driven way. We focused on the constraints of the mobile-device scenario, with its data acquisition peculiarities and hardware limitations.

By using a powerful but lightweight deep convolutional architecture as a foundation, our data-driven approach lets us focus on the problem definition itself instead of being tied to specific handcrafted features. In the first method, we show how to adapt this architecture to classify images as genuine or attack, and train a deep model using aligned whole-face images as input, much like the majority of existing PAD pipelines. In the second method, we leverage the available training data and the fully-convolutional architecture, showing how to train the network with face patches of varying detail, increasing the number of available training examples, encouraging the model to be robust to changes in resolution, and avoiding overfitting to specific combinations of facial features. In the third method, we build upon the previous idea, but further reformulate the problem by training with a multi-objective loss function that encourages real-access examples from the same devices to be more compactly located in the learned feature spaces, while also reducing inter-device confusion.

In the context of this work, we also collected a novel face PAD dataset targeted at the mobile-device scenario. RECOD-MPAD contains videos from 45 users captured with 2 modern smartphones in 5 sessions, covering low-light and outdoor lighting scenarios, with much higher intra and inter-session variability than in existing public datasets. For evaluating PAD methods, we defined not only an overall protocol considering all available data, but also more challenging factor-disjoint protocols.

Our data-driven approach proved to be superior to a recent competitive handcrafted baseline, as well as a similar CNN with pre-trained features. This suggests that, contrary to popular belief, complex deep learning models can generalize better than handcrafted alternatives, even when trained with arguably limited amounts of data. Crucially, the trained architecture has very small memory requirements, and can make predictions within a fraction of a second in modern smartphones. This validates the potential of data-driven approaches in solving the presentation-attack detection problem in mobile environments.

By evaluating the data-driven models under RECOD-MPAD's factor-disjoint protocols, in which the testing sets retain unseen sessions, attack varieties, or user devices, we

were able to test the limits of software-based PAD methods. Despite the generalization challenges, our patches-based methods performed well in most cases, but degradation in performance is significant when compared to evaluations in which the only disjoint factor is the user identity. More specifically, the results highlight the importance of training with diverse illumination scenarios – performance for unseen extreme low-light conditions is particularly bad. In the cross-attack protocol, we observed the greatest difficulty in generalization when dealing with unseen attacks from the larger display. In the cross-device protocol, generalization was better when training with the user device that captures sharper images, although we highlight the significant advantage of using the method trained with the proposed loss function in this case, which suggests the benefit goes beyond separating devices during training. These insights are useful both for motivating future PAD algorithms, and for the construction of new training and evaluation datasets.

We also proposed a computationally cheap on-device procedure that can be used to find user-specific decision thresholds, improving a model's usefulness in the operational phase. By taking advantage of the diverse sessions in RECOD-MPAD, we showed that error rates can effectively be reduced in most cases, even when considering a reference gallery of only 2 sessions. At the very least, the procedure can make a model operate at its best, constrained to a controlled user-inconvenience level.

Additional evaluations with the OULU-NPU dataset – used as benchmark in a competition in the year 2017 – demonstrated how our methods compare to the state of the art in another standardized setting. Even though our models were not particularly tuned for this dataset, the results are promising when compared to the top-performing participants. A simple score-fusion strategy further improved performance in most cases, suggesting some degree of complementarity among the models. More importantly, the three proposals surpassed all but the first place in the most challenging protocol of the competition, with Method III performing particularly well.

Our models demonstrated promising results in terms of error rates, when compared to the state of the art, but we call attention to the important fact that single software-based PAD methods are still not good enough for real-world security requirements. As of now, the biggest challenge is in limitations of datasets available for training and evaluating models. As the size and quality of these datasets improve, we expect to see a proportional improvement in the accuracy and usefulness of data-driven models.

We realize not everything can fit in a single work, and many more questions than can be answered arise during a researcher's journey. As future work, we can suggest a few directions. More immediately, one could systematically study the complementarity and fusion potential of the proposed methods to existing PAD methods. Another promising path is in evaluating the effect of using other architectures, but care must be taken with regards to the constraints of the target scenario. Finally, we hypothesize the proposed multi-objective loss would perform particularly well when training with multiple more heterogeneous datasets, but the question is still open.

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] André Anjos, Murali Mohan Chakka, and Sébastien Marcel. Motion-based counter-measures to photo attacks in face recognition. *IET Biometrics*, 3(3):147–158, 2013.

[3] André Anjos and Sébastien Marcel. Counter-measures to photo attacks in face recognition: a public database and a baseline. In *IEEE International Joint Conference on Biometrics*, pages 1–7, 2011.

[4] Shervin Rahimzadeh Arashloo, Josef Kittler, and William Christmas. An anomaly detection approach to face spoofing detection: A new formulation and evaluation protocol. *IEEE Access*, 5:13868–13882, 2017.

[5] Jiamin Bai, Tian-Tsong Ng, Xinting Gao, and Yun-Qing Shi. Is physics-based liveness detection truly possible with a single image? In *IEEE International Symposium on Circuits and Systems*, pages 3425–3428, 2010.

[6] Wei Bao, Hong Li, Nan Li, and Wei Jiang. A liveness detection method for face recognition based on optical flow field. In *International Conference on Image Analysis and Signal Processing*, pages 233–236, 2009.

[7] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[8] Samarth Bharadwaj, Tejas I. Dhamecha, Mayank Vatsa, and Richa Singh. Computationally efficient face spoofing detection with motion magnification. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 105–110, 2013.

[9] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual Workshop on Computational Learning Theory*, pages 144–152, 1992.

[10] Z. Boulkenafet, J. Komulainen, Z. Akhtar, A. Benlamoudi, S. Bekhouche, A. Ouafi, F. Dornaika, A. Taleb-Ahmed, L. Qin, F. Peng, L.B. Zhang, M. Long, S. Bhilare, V. Kanhangad, A. Costa-Pazo, E. Vazquez-Fernandez, D. Perez-Cabo, J. J. Moreira-Perez, D. Gonzalez-Jimenez, A. Mohammadi, S. Bhattacharjee, S. Marcel, S. Volkova, Y. Tang, N. Abe, L. Li, X. Feng, Z. Xia, X. Jiang, S. Liu, R. Shao, P. C. Yuen, W. Almeida, F. Andalo, R. Padilha, G. Bertocco, W. Dias, J. Wainer, R. Torres, A. Rocha, M. A. Angeloni, G. Folego, A. Godoy, and A. Hadid. A competition on generalized software-based face presentation attack detection in mobile scenarios. In *IEEE International Joint Conference on Biometrics*, 2017.

[11] Z. Boulkenafet, J. Komulainen, L. Li, X. Feng, and A. Hadid. Oulu-npu: A mobile face presentation attack database with real-world variations. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 612–618, 2017.

[12] Zinelabidine Boulkenafet, Jukka Komulainen, and Abdenour Hadid. Face anti-spoofing based on color texture analysis. In *IEEE International Conference on Image Processing*, pages 2636–2640, 2015.

[13] Zinelabidine Boulkenafet, Jukka Komulainen, and Abdenour Hadid. Face spoofing detection using colour texture analysis. *IEEE Transactions on Information Forensics and Security*, 11(8):1818–1830, 2016.

[14] Murali Mohan Chakka, Andre Anjos, Sebastien Marcel, Roberto Tronci, Daniele Muntoni, Gianluca Fadda, Maurizio Pili, Nicola Sirena, Gabriele Murgia, Marco Ristori, et al. Competition on counter measures to 2-d facial spoofing attacks. In *IEEE International Joint Conference on Biometrics*, pages 1–6, 2011.

[15] Girija Chetty and Michael Wagner. Liveness detection using cross-modal correlations in face-voice person authentication. In *ISCA European Conference on Speech Communication and Technology*, pages 2181–2184, 2005.

[16] Ivana Chingovska, André Anjos, and Sébastien Marcel. On the effectiveness of local binary patterns in face anti-spoofing. In *IEEE International Conference of the Biometrics Special Interest Group*, pages 1–7, 2012.

[17] Ivana Chingovska, Jinwei Yang, Zhen Lei, Dong Yi, Stan Z. Li, Olga Kahm, Christian Glaser, Naser Damer, Arjan Kuijper, Alexander Nouak, et al. The 2nd competition on counter measures to 2d face spoofing attacks. In *IAPR International Conference on Biometrics*, pages 1–6, 2013.

[18] Tanzeem Choudhury, Brian Clarkson, Tony Jebara, and Alex Pentland. Multimodal person recognition using unconstrained audio and video. In *International Conference on Audio- and Video-Based Biometric Person Authentication*, pages 176–181, 1999.

[19] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[20] Artur Costa-Pazo, Sushil Bhattacharjee, Esteban Vazquez-Fernandez, and Sebastien Marcel. The replay-mobile face presentation-attack database. In *IEEE International Conference of the Biometrics Special Interest Group*, pages 1–7, 2016.

[21] David Cox and Nicolas Pinto. Beyond simple features: A large-scale feature search approach to unconstrained face recognition. In *International Workshop on Automatic Face and Gesture Recognition*, pages 8–15, 2011.

[22] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, 1989.

[23] Tiago de Freitas Pereira, André Anjos, José Mario De Martino, and Sébastien Marcel. Can face anti-spoofing countermeasures work in a real world scenario? In *IAPR International Conference on Biometrics*, pages 1–8, 2013.

[24] Tiago de Freitas Pereira, Jukka Komulainen, André Anjos, José Mario De Martino, Abdenour Hadid, Matti Pietikäinen, and Sébastien Marcel. Face liveness detection using dynamic texture. *EURASIP Journal on Image and Video Processing*, 2014(1):2, 2014.

[25] Maria De Marsico, Michele Nappi, Daniel Riccio, and Jean-Luc Dugelay. Moving face spoofing detection via 3d projective invariants. In *IAPR International Conference on Biometrics*, pages 73–78, 2012.

[26] Tejas I. Dhamecha, Aastha Nigam, Richa Singh, and Mayank Vatsa. Disguise detection and face recognition in visible and thermal spectrums. In *IAPR International Conference on Biometrics*, pages 1–8, 2013.

[27] Nesli Erdogmus and Sébastien Marcel. Spoofing in 2d face recognition with 3d masks and anti-spoofing with kinect. In *IEEE International Conference on Biometrics: Theory, Applications, and Systems*, pages 1–6, 2013.

[28] Javier Galbally, Sébastien Marcel, and Julian Fierrez. Biometric antispoofing methods: A survey in face recognition. *IEEE Access*, 2:1530–1552, 2014.

[29] Javier Galbally, Sébastien Marcel, and Julian Fierrez. Image quality assessment for fake biometric detection: Application to iris, fingerprint, and face recognition. *IEEE Transactions on Image Processing*, 23(2):710–724, 2014.

[30] Diego Gragnaniello, Giovanni Poggi, Carlo Sansone, and Luisa Verdoliva. An investigation of local descriptors for biometric spoofing detection. *IEEE Transactions on Information Forensics and Security*, 10(4):849–863, 2015.

[31] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

[32] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and < 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[33] Information technology – Biometric presentation attack detection – Part 3: Testing and reporting (ISO/IEC 30107-3:2017). Standard, International Organization for Standardization, Geneva, CH, September 2017.

[34] Anil K. Jain, Arun Ross, and Salil Prabhakar. An introduction to biometric recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(1):4–20, 2004.

[35] Gahyun Kim, Sungmin Eum, Jae Kyu Suhr, Dong Ik Kim, Kang Ryoung Park, and Jaihie Kim. Face liveness detection based on texture and frequency analyses. In *IAPR International Conference on Biometrics*, pages 67–72, 2012.

[36] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[37] Klaus Kollreider, Hartwig Fronthaler, and Josef Bigun. Evaluating liveness by face images and the structure tensor. In *IEEE Workshop on Automatic Identification Advanced Technologies*, pages 75–80, 2005.

[38] Klaus Kollreider, Hartwig Fronthaler, and Josef Bigun. Verifying liveness by multiple experts in face biometrics. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–6, 2008.

[39] Klaus Kollreider, Hartwig Fronthaler, and Josef Bigun. Non-intrusive liveness detection by face images. *Image and Vision Computing*, 27(3):233–244, 2009.

[40] Jukka Komulainen, Abdenour Hadid, and Matti Pietikäinen. Face spoofing detection using dynamic texture. In *Asian Conference on Computer Vision*, pages 146–157, 2012.

[41] Jukka Komulainen, Abdenour Hadid, Matti Pietikäinen, André Anjos, and Sébastien Marcel. Complementary countermeasures for detecting scenic face spoofing attacks. In *IAPR International Conference on Biometrics*, pages 1–7, 2013.

[42] Neslihan Kose and Jean-Luc Dugelay. Reflectance analysis based countermeasure technique to detect face mask attacks. In *IEEE International Conference on Digital Signal Processing*, pages 1–6, 2013.

[43] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[44] Andrea Lagorio, Massimo Tistarelli, Marinella Cadoni, Clinton Fookes, and Sridha Sridharan. Liveness detection based on 3d face shape analysis. In *International Workshop on Biometrics and Forensics*, pages 1–4, 2013.

[45] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[46] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems*, pages 396–404, 1990.

[47] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[48] Jiangwei Li, Yunhong Wang, Tieniu Tan, and Anil K. Jain. Live face detection based on the analysis of fourier spectra. In *Biometric Technology for Human Identification*, volume 5404, pages 296–304, 2004.

[49] Lei Li, Xiaoyi Feng, Zinelabidine Boulkenafet, Zhaoqiang Xia, Mingming Li, and Abdenour Hadid. An original face anti-spoofing approach using partial convolutional neural network. In *International Conference on Image Processing Theory Tools and Applications*, pages 1–6, 2016.

[50] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[51] Jukka Määttä, Abdenour Hadid, and Matti Pietikäinen. Face spoofing detection from single images using micro-texture analysis. In *IEEE International Joint Conference on Biometrics*, pages 1–7, 2011.

[52] Jukka Määttä, Abdenour Hadid, and Matti Pietikäinen. Face spoofing detection from single images using texture and local shape analysis. *IET Biometrics*, 1(1):3–10, 2012.

[53] David Menotti, Giovani Chiachia, Allan Pinto, William Robson Schwartz, Helio Pedrini, Alexandre Xavier Falcão, and Anderson Rocha. Deep representations for iris, face, and fingerprint spoofing detection. *IEEE Transactions on Information Forensics and Security*, 10(4):864–879, 2015.

[54] Timo Ojala, Matti Pietikainen, and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.

[55] Gang Pan, Lin Sun, Zhaohui Wu, and Shihong Lao. Eyeblink-based anti-spoofing in face recognition from a generic webcamera. In *IEEE International Conference on Computer Vision*, pages 1–8, 2007.

[56] Gang Pan, Lin Sun, Zhaohui Wu, and Yueming Wang. Monocular camera-based face liveness detection by combining eyeblink and scene context. *Telecommunication Systems*, 47(3-4):215–225, 2011.

[57] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In *British Machine Vision Conference*, pages 41.1–41.12, 2015.

[58] Keyurkumar Patel, Hu Han, and Anil K. Jain. Cross-database face antispoofing with robust feature representation. In *Chinese Conference on Biometric Recognition*, pages 611–619, 2016.

[59] Keyurkumar Patel, Hu Han, Anil K. Jain, and Greg Ott. Live face video vs. spoof face video: Use of moiré patterns to detect replay video attacks. In *IAPR International Conference on Biometrics*, pages 98–105, 2015.

[60] Bruno Peixoto, Carolina Michelassi, and Anderson Rocha. Face liveness detection under bad illumination conditions. In *IEEE International Conference on Image Processing*, pages 3557–3560, 2011.

[61] Allan Pinto, Helio Pedrini, William Robson Schwartz, and Anderson Rocha. Face spoofing detection through visual codebooks of spectral temporal cubes. *IEEE Transactions on Image Processing*, 24(12):4726–4740, 2015.

[62] Allan Pinto, William Robson Schwartz, Helio Pedrini, and Anderson de Rezende Rocha. Using visual rhythms for detecting video-based facial spoof attacks. *IEEE Transactions on Information Forensics and Security*, 10(5):1025–1038, 2015.

[63] Nicolas Pinto, David Doukhan, James J. DiCarlo, and David D. Cox. A high-throughput screening approach to discovering good forms of biologically inspired visual representation. *PLOS Computational Biology*, 5(11):e1000579, 2009.

[64] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[65] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[66] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[67] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.

[68] William Robson Schwartz, Anderson Rocha, and Helio Pedrini. Face spoofing detection through partial least squares and low-level descriptors. In *IEEE International Joint Conference on Biometrics*, pages 1–8, 2011.

[69] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 806–813, 2014.

[70] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[71] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[72] Holger Steiner, Andreas Kolb, and Norbert Jung. Reliable face anti-spoofing using multispectral swir imaging. In *IAPR International Conference on Biometrics*, pages 1–8, 2016.

[73] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[74] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, 2014.

[75] Xiaoyang Tan, Yi Li, Jun Liu, and Lin Jiang. Face liveness detection from a single image with sparse low rank bilinear discriminative model. *European Conference on Computer Vision*, pages 504–517, 2010.

[76] Di Wen, Hu Han, and Anil K. Jain. Face spoof detection with image distortion analysis. *IEEE Transactions on Information Forensics and Security*, 10(4):746–761, 2015.

[77] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.

[78] Junjie Yan, Zhiwei Zhang, Zhen Lei, Dong Yi, and Stan Z Li. Face liveness detection by exploring multiple scenic clues. In *IEEE International Conference on Control Automation Robotics and Vision*, pages 188–193, 2012.

[79] Jianwei Yang, Zhen Lei, and Stan Z. Li. Learn convolutional neural network for face anti-spoofing. *arXiv preprint arXiv:1408.5601*, 2014.

[80] Zhiwei Zhang, Junjie Yan, Sifei Liu, Zhen Lei, Dong Yi, and Stan Z. Li. A face antispoofing database with diverse attacks. In *IAPR International Conference on Biometrics*, pages 26–31, 2012.

[81] Zhiwei Zhang, Dong Yi, Zhen Lei, and Stan Z. Li. Face liveness detection by learning multispectral reflectance distributions. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 436–441, 2011.

[82] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929, 2016.