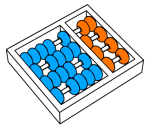
Universidade de Campinas
Instituto de Computação



Rafael Soares Padilha

Two-tiered facial verification for mobile devices

Verificação facial em duas etapas para dispositivos móveis

CAMPINAS
2017

Rafael Soares Padilha

Two-tiered facial verification for mobile devices

Verificação facial em duas etapas para 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. Jacques Wainer
Co-supervisor/Coorientadora: Dra. Fernanda Alcântara Andaló

Este exemplar corresponde à versão final da Dissertação defendida por Rafael Soares Padilha e orientada pelo Prof. Dr. Jacques Wainer.

CAMPINAS
2017

Agência(s) de fomento e nº(s) de processo(s): Não se aplica.

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

P134t

Padilha, Rafael Soares, 1991-
Two-tiered facial verification for mobile devices / Rafael Soares Padilha. – Campinas, SP : [s.n.], 2017.

Orientador: Jacques Wainer.
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. Wainer, Jacques,1958-. II. Andaló, Fernanda Alcântara,1981-. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: Verificação facial em duas etapas para 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:
Jacques Wainer [Orientador]
Sandra Eliza Fontes de Avila
Aparecido Nilceu Marana
Data de defesa: 01-09-2017
Programa de Pós-Graduação: Ciência da Computação

Universidade de Campinas
Instituto de Computação

Rafael Soares Padilha

Two-tiered facial verification for mobile devices

Verificação facial em duas etapas para dispositivos móveis

Banca Examinadora:

- Prof. Dr. Jacques Wainer (Supervisor)
  Instituto de Computação - UNICAMP
- Prof. Dra. Sandra Eliza Fontes de Avila
  Instituto de Computação - UNICAMP
- Prof. Dr. Aparecido Nilceu Marana
  Faculdade de Ciências - UNESP

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

Campinas, 01 de setembro de 2017

Acknowledgements

No research can be done without the support, directly or indirectly, from many people. These individuals either pave the road before the researcher goes through it — in the form of previous scientific works and technologies — or walk alongside with him or her — as mentors, fellow researchers, family and friends. Both are equally important, however here we pay our acknowledgements to the latter group, those in direct contact with us throughout this work.

First and foremost, I would like to thank my parents, Reinaldo and Mara, for everything they have done for me. They have taught me by example what is to be a decent human being, putting effort and love into everything I do! I am very lucky to have such amazing parents backing me up! We joke at home that a piece of my degree belongs to them and, obviously, that could not be more true.

I would also like to thank my supervisors (Jacques Wainer, Fernanda Andaló, Anderson Rocha and Ricardo Torres); their insights, advices, feedback and support were essential not only to the success of this research but also to lead me in my initial steps through scientific research! No less important are the other members of my research group. Waldir, William and Gabriel, I am grateful for all the help and time we have spent together both inside and outside of the laboratory. To all members of BioLive, thank you! I am glad there were others with me walking this path through the ups and downs of it!

I would like to express my appreciation to all my other friends; the ones that live with me, those I can see frequently and those that I can not. They all had their share of me through all the phases of this work.

Finally, I would like to thank the support from Motorola — not only financially but also with the huge help from Benicio Goulart and Thiago Resek — and from UNICAMP (it has been more than 8 years with you... and counting).

Abstract

Mobile devices, such as smartphones and tablets, had their popularity and affordability greatly increased in recent years. As a consequence of their ubiquity, these devices now carry all sorts of personal data (e.g., photos, text conversations, GPS coordinates, banking information) that should be accessed only by the device owner. Even though knowledge-based procedures, such as entering a PIN or drawing a pattern, are still the main methods to secure the owner's identity, recently biometric traits have been employed for a more secure and effortless authentication. Among them, face recognition has gained more attention in past years due to recent improvements in image-capturing devices and the availability of images in social networks. In addition to that, the increase in computational resources, with multiple CPUs and GPUs, enabled the design of more complex and robust models, such as deep neural networks. Although the capabilities of mobile devices have been growing in past years, most recent face recognition techniques are still not designed considering the mobile environment's characteristics, such as limited processing power, unstable connectivity and battery consumption. In this work, we propose a facial verification method optimized to the mobile environment. It consists of a two-tiered procedure that combines hand-crafted features (histogram of oriented gradients and local region principal component analysis) and a convolutional neural network to verify if the person depicted in a picture corresponds to the device owner. We also propose *Hybrid-Fire Convolutional Neural Network*, an architecture tweaked for mobile devices that process encoded information of a pair of face images. Finally, we expose a technique to adapt our method's acceptance thresholds to images with different characteristics than those present during training, by using the device owner's enrolled gallery. The proposed solution performs a par to the state-of-the-art face recognition methods, while having a model 16 times smaller and 4 times faster when processing an image in recent smartphone models. Finally, we have collected a new dataset of selfie pictures comprising 2873 images from 56 identities with varied capture conditions, that hopefully will support future researches in this scenario.

Resumo

Dispositivos móveis, como smartphones e tablets, se tornaram mais populares e acessíveis nos últimos anos. Como consequência de sua ubiquidade, esses aparelhos guardam diversos tipos de informações pessoais (fotos, conversas de texto, coordenadas GPS, dados bancários, entre outros) que só devem ser acessadas pelo dono do dispositivo. Apesar de métodos baseados em conhecimento, como senhas numéricas ou padrões, ainda estarem entre as principais formas de assegurar a identidade do usuário, traços biométricos têm sido utilizados para garantir uma autenticação mais segura e prática. Entre eles, reconhecimento facial ganhou atenção nos últimos anos devido aos recentes avanços nos dispositivos de captura de imagens e na crescente disponibilidade de fotos em redes sociais. Aliado a isso, o aumento de recursos computacionais, com múltiplas CPUs e GPUs, permitiu o desenvolvimento de modelos mais complexos e robustos, como redes neurais profundas. Porém, apesar da evolução das capacidades de dispositivos móveis, os métodos de reconhecimento facial atuais ainda não são desenvolvidos considerando as características do ambiente móvel, como processamento limitado, conectividade instável e consumo de bateria. Neste trabalho, apresenta-se um método de verificação facial otimizado para o ambiente móvel. Ele consiste em um procedimento em dois níveis que combina engenharia de características (histograma de gradientes orientados e análise de componentes principais por regiões) e uma rede neural convolucional para verificar se o indivíduo presente em uma imagem corresponde ao dono do dispositivo. Também propõe-se a *Hybrid-Fire Convolutional Neural Network*, uma arquitetura ajustada para dispositivos móveis que processa informação de pares de imagens. Finalmente, é apresentada uma técnica para adaptar o limiar de aceitação do método proposto para imagens com características diferentes daquelas presentes no treinamento, utilizando a galeria de imagens do dono do dispositivo. A solução proposta se compara em acurácia aos métodos de reconhecimento facial do estado da arte, além de possuir um modelo 16 vezes menor e 4 vezes mais rápido ao processar uma imagem em smartphones modernos. Por último, foi organizada uma base de dados composta por 2873 selfies de 56 identidades capturadas em condições diversas, a qual esperamos que ajude pesquisas futuras realizadas neste cenário.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

The need to secure one's identity is present in a variety of everyday activities [42, 91], such as allowing or denying access to a requested service, a place, or sensitive information. Examples of these include ensuring the identity of a voter during an election, access-control to work environments and bank accounts. Traditional methods, including the ones based on knowledge (e.g., keywords, secret question) or based on tokens (e.g., smart cards), might be ineffective as they can be shared, lost, stolen or manipulated with ease.

In this sense, several systems use biometric traits to secure the identity of an individual [42]. These traits can be any human biological and/or behavioral characteristic capable of uniquely identifying a person [43, 59, 90]. Examples of these include face traces, voice, fingerprints, ear shape, hand geometry, iris, gait, keystroke, and infrared veins thermogram of hand or face [21].

Biometric systems work in one of two different tasks: verification and identification [42]. Verification, or authentication, is to verify a person's claimed identity, i.e., the authentication of a person is performed by reading and comparing the input biometric identifier captured by an acquisition sensor (query) with the biometric identifier of the same person previously stored in a database (template). The comparison between the query and the template is performed by a matching algorithm, which produces a similarity score used to decide whether or not the access should be granted to the user. Identification is concerned with identifying a person by comparing the input biometric identifier with a database of previously known identifiers and their respective owners.

Considering the importance of the information or service in question, a practical biometric system is designed to be fast, accurate, easy to use, acceptable by the intended population, and robust to attacks and fraudulent methods [41, 43]. They also have to deal with a variety of problems [21], such as:

- Noisy captured data: either resulted by an external factor (voice altered by cold or eyes covered by sunglasses, for example) or a defective sensor;
- Intra-class variations: variations caused by an interaction with a sensor or due to external conditions for the same user;
- Inter-class similarities: similarities in the features used to represent a biometric trait between a large range of different users;

- Non-universality: it may be difficult to extract a particular biometric trait from a user. For example, people with irregular ridges using a fingerprint-based system.

Among the different biometric modalities, face recognition is a very important one [51]. During the past years, it has gained more attention with improvements regarding quality, affordability and ubiquity of image-capturing devices (surveillance cameras, mobile phone cameras), the many possible commercial uses, not only in security (e.g., to authenticate the smartphone owner in payments [25, 61, 62, 65]), but in other areas such as entertainment (e.g., video games and human-computer interaction) [96] and the huge amount of images available in social networks. Boosted by this large volume of images, powerful statistical models that otherwise struggled with the lack of data have not only become viable, but also were able to improve the robustness of visual systems to noise and variation, such as illumination, pose and occlusion [82]. In this work, we build upon one of such models: deep neural network, which has already been applied in the face recognition pipeline for tasks like face detection [63], alignment [78] and verification [16, 38].

With mobile and wearable devices becoming cheaper and more popular [6, 60], face recognition systems are being integrated into them in a wide range of tasks. A growing number of applications uses face recognition to analyze and interpret the user's actions, intentions and/or behaviors, acting according to some person's preferences or state of mind [67]. For example, through facial expressions, it is possible to identify if the user is confused, happy or impatient and take that into account when presenting information on the screen.

Although the capabilities of mobile devices have been growing in past years, it is necessary to bear in mind their limitations when designing applications for them [14]. They have limited processing power that may not be sufficient to run many complex vision and pattern recognition algorithms, as well as a small memory space that may not be suitable to store several face features of high dimensionality. Despite all these resource limitations, we still desire to have a fast and accurate face recognition system, that does not consume too much energy, since these devices run in low-powered batteries [14].

Nonetheless, research and design aimed at the mobile environment are not only about limitations and drawbacks; it also has some unique characteristics that can be explored when implementing such a system. For example, unlike some technologies, mobile devices are usually single-user which means that, even though usage behavior depends on the owner profile [24], in most cases we can approach face recognition as a verification task, rather than identification. We could also leverage from this by collecting new face pictures from its user regularly, improving the system's ability to recognize the device owner.

In this work, we design a set of techniques for verification on mobile devices that uses *selfie* images to authenticate, i.e., self-portrait pictures usually taken with a smartphone camera that have become very popular in past years. Fig. 1.1 shows examples of selfies and possible variations in illumination, pose and occlusion present in them. Alongside with a deep learning approach, we use traditional techniques as complementary tools, and also as a way to extract specific information of the device owner.

Figure 1.1: Examples of selfies reproduced from Flickr with different acquisition conditions regarding illumination, partial occlusion, head pose and alterations due to physical changes, e.g., new hairstyle or make-up.

1.1 Research questions

The main research questions that guide us throughout this work are:

I. Recently, Deep Learning has become the state-of-art for face recognition, surpassing traditional feature engineered methods. However complex deep networks are computationally expensive, while the other tends to be fast and memory efficient.

   (i) *Considering the mobile environment's resource limitations and the computational cost of running a deep network in it, is deep learning a necessary approach for this kind of application?*

   (ii) *Is it possible to design a deep learning solution for the face verification problem bearing in mind the environment's characteristics?*

   (iii) *Would a fusion of deep and hand-crafted approaches lead to better accuracy than the methods separately?*

II. The more face images from the device owner we have in the gallery, the more information we will have to authenticate or reject a new picture.

   (i) *How many images should the gallery have?*

   (ii) *In what resolution should these images be?*

III. Most recent face recognition solutions address the identification scenario, usually with multi-class deep networks. *Is it possible to adapt multi-class networks for the binary verification scenario, without a significant increase in memory and processing time?*

IV. Since mobile devices are usually single-user, *would user-specific information improve our solution?*

    (i) *If so, how should we incorporate it to our method?*

1.2 Contributions

This master's thesis introduces a number of contributions to different aspects of facial recognition and deep learning targeted to the mobile environment:

- We propose a facial authentication method that combines hand-crafted and deep learning features. The process consists of a two-tier solution based on a set of user-specific classifiers trained locally in the mobile device and on a group of pre-trained classifiers to determine if two face pictures belong to the same person or not.
- We present a mobile-tweaked Convolutional Neural Network (CNN) architecture, adapted from VGG network [64], resulting in a model up to 16 times smaller and 4 times faster than VGG. Besides the architectural details, we also discuss the decisions taken during its design to aid reproducibility.
- As a way to better adapt our method to images with different characteristics that those present during training, we propose a technique to automatically learn the acceptance threshold of a classifier of our solution based on the face images provided by the user.
- A public dataset composed of selfie pictures with different acquisition conditions regarding illumination and head pose. The dataset comprises 56 identities and 2873 images, and is one of the first in literature to focus on selfies for authentication.

1.3 Thesis Organization

The face recognition problem has been extensively addressed in different perspectives and approaches and is one of the most active topics of interest in computer vision [12]. However only recently there has been an effort to propose techniques that take into account the mobile environment's characteristics. Therefore, in order to better contextualize this thesis, in Chapter 2 we present a summary about the methods and techniques that we based this work on, while also discussing previous researches, pointing out their characteristics and relevance to the proposed system.

We describe in Chapter 3 the dataset proposed for this research, comparing them to others available in the literature, and also how we evaluated our results. In Chapter 4, we go in depth with each key aspect of the proposed solution. We discuss in Chapter 5 the experimental results and the impact of each component presented in Chapter 4. We also compare the solution with state-of-the-art methods and analyze their performance on the proposed dataset. Finally, we compile the contributions and experimental findings of this thesis in Chapter 6, outlining new directions to guide this work in the future.

Chapter 2

Background

Several psychophysics and neuroscience studies have been trying to answer questions related to face recognition [96], such as *"Is face recognition a dedicated process?"*, or *"Is face perception the result of holistic or region-based analysis?"*. This kind of question is important when studying and proposing computational methods for face detection and recognition. In fact, hints to answer them can be observed in the literature by the study of holistic, region-based, and hybrid approaches. Before diving into the proposed method, it is important to explore the concepts upon which we built our solution. In this chapter, we begin by presenting a general authentication pipeline, as well as some of the traditional techniques based on hand-crafted features applied to this task. We follow with a discussion about state-of-the-art researches using CNNs to tackle this problem and the efforts to integrate these solutions into the mobile environment.

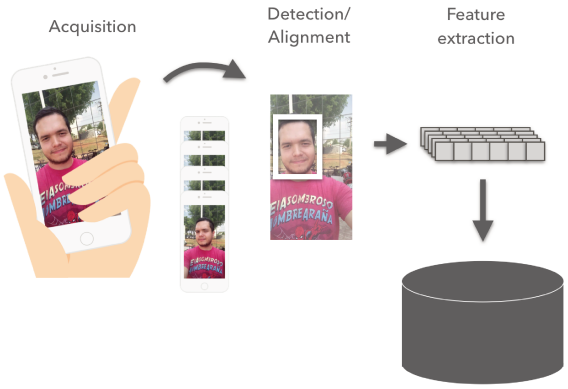
2.1 Authentication pipeline

To perform face authentication in mobile devices, besides taking into consideration resource limitations, it is necessary to consider two tasks: face detection, which consists of localizing a face in an image; and face verification, which is, given an image containing a face, determining if they belong to the device owner.

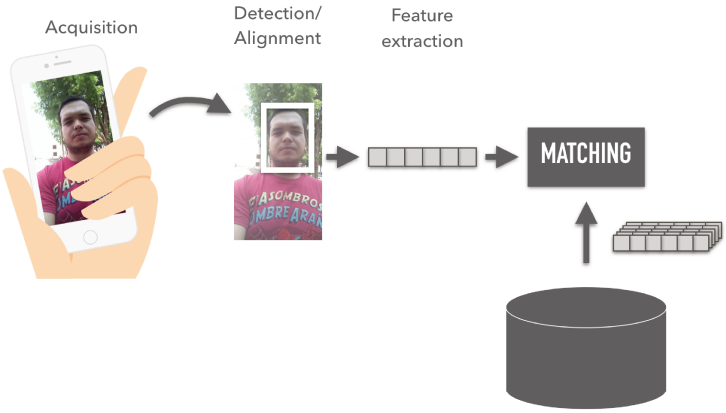
A face verification pipeline consists of two main modules: enrollment, which is the acquisition of biometric features (in this case, images of the face) of a user; and the authentication itself, which compares the current acquired biometric trait with the ones previously acquired, and determines if they belong to the same person.

In the enrollment process (Fig. 2.1a), the user acquire one or more images with the mobile camera, then the system detects the face in each image, and extracts and stores features of each face. In the authentication process (Fig. 2.1b), the user, who wants to have access to the mobile phone, uses the mobile camera to acquire one image of one's face, then the system detects the face in the image, extracts the same type of features of the face, and performs a matching algorithm to compare the incoming feature vector with the ones previously stored, granting access if the person is the owner.

Automatic face detection is the foundation of all applications revolving around facial image analysis. The literature on face detection is rich and dates back 50 years. Nowadays,



(a) During the enrollment, the face is detected in each acquired image and its features are extracted and stored.



(b) During the authentication, the detected face has the same type of features extracted and the system compares them with previously stored feature vectors to determine if the captured person is the owner.

Figure 2.1: Pipeline for (a) enrollment and (b) authentication on mobile

face detection algorithms can be appropriately applied in real-world settings, even in constrained devices. Due to the copious availability of detection methods and the high probability of user cooperation, our research efforts are not focused towards this matter. Instead, we assume that the image already had its face detected before normalization.

During the normalization step (Fig. 2.2), the goal is to bring face images to a common scale or into alignment, thus increasing the cohesion among previous and future inputs, even in variably acquisition conditions. For instance, face images can be cropped and aligned, and some of its properties, such as illumination, may be altered. In the proposed method, we crop and align face images using the same method of several works [56, 68]. The location of the eyes in each image is fixed on standard pixel locations, i.e., the distance of the center of the eyes to the image boundaries is the same for all images.

In the feature extraction step, we compute a representation of the face image, different from the image space, where the relevant discriminative information is captured, while redundant information is discarded, facilitating subsequent learning steps.

In this project, we studied and tested several feature extraction approaches, from

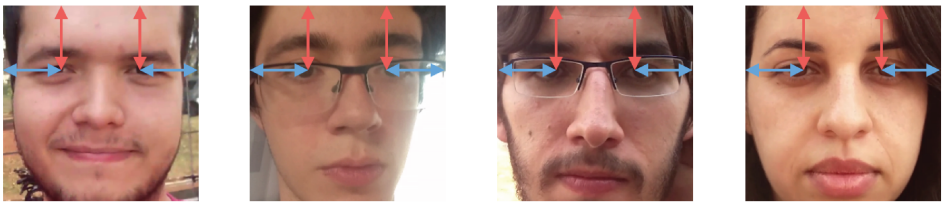


Figure 2.2: Face alignment.

hand-crafted ones to directly learning useful representations from image pixels through deep CNNs. In Sections 2.2 and 2.3, we discuss relevant works on both ends.

2.2 Hand-crafted features

Hand-crafted features are designed using domain knowledge of the data to create representations of face images in a process called description or feature engineering.

In 1973, Kanade proposed a method [45] to automatically detect a face by representing it with angles and distances of fiducial points, such as eye corners and mouth extrema. However, methods based only on geometrical features are now known to discard important information of the face appearance that could be used to identify a person.

In 1991, Turk and Pentland presented Eigenfaces [85], an holistic method that uses Principal Component Analysis (PCA) for learning a low dimensional subspace of face representations, whereto faces are projected before recognition. In a similar line, Fisherfaces [2] incorporates labels to the learning procedure and performs a Linear Discriminant Analysis (LDA) to find a subspace that maximizes the ratio between-class and within-class variance.

An extension to Kanade approach was the use of the response of Gabor filters to represent appearance over the regions around fiducial points [92]. By incorporating appearance and locally representing facial features, this method inspired a large range of techniques. Following this path, Ahonen *et al.* [1] proposed the use of Local Binary Pattern (LBP) texture features to describe local regions in a dense grid on the face, computing a histogram for each region that are further combined to compose the face representation.

Although hand-crafted features achieved promising results in the constrained scenario — with little variation in illumination, pose, facial expression and occlusion — they lack performance in the unconstrained one, and several researches combine them together to increase robustness [19, 54, 83].

Two hand-crafted features have special importance to this work: Histogram of Oriented Gradients (HOG) and Local Region Principal Component Analysis (LRPCA).

2.2.1 Histogram of Oriented Gradients

Dalal and Triggs [20] explored the use gradients features for human detection in images. Although similar features have been proposed [26, 55], this method differs by evaluating normalized local histograms of image gradient orientations in a dense grid.

The basic idea is that local object appearance and shape can often be characterized by the distribution of local intensity gradients. In practice, the image is divided into small cells, and, for each cell, a local histogram of gradient directions over the pixels is accumulated. The combined histograms are used to represent the face image (Fig. 2.3). For better invariance, a measure of local histogram energy is accumulated over larger spatial regions (blocks) and the results are used to normalize all cells in the block.

In this work, we use 128x128 resolution gray-scale images divided in 16x16 cells and 32x32 blocks with stride of 16. The output HOG feature vector has size 1764.

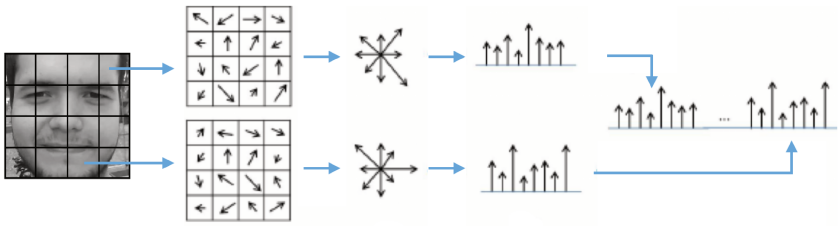


Figure 2.3: HOG descriptor outline, adapted from [11].

2.2.2 Local Region Principal Component Analysis

Phillips *et al.* [68] extended Eigenfaces [85] by calculating a low dimensional projection space through PCA for the whole face and 13 local regions within it, centered relative to the average location of the eyes, eyebrows, nose and mouth, as illustrated in Fig. 2.4.

The face image is aligned using the eyes' position, normalized to weaken illumination variations and its pixel values are adjusted to have a sample mean of zero and a sample standard deviation of one. During training, the PCA subspaces are constructed for each region, retaining only part of the eigenvectors. Whereas in testing, a face image has its regions extracted and subsequently projected to the respective PCA subspace. The outputs of each region are concatenated to form the final LRPCA feature vector.

In our method, instead of the 14 regions, we construct PCA subspaces for two 36x36 regions centered around each eye and a region comprising the whole 128x128 face image, retaining 92% of the variance of the training data. The projections over these three

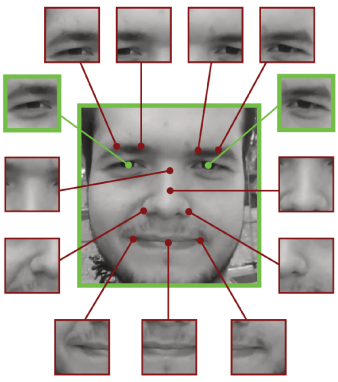


Figure 2.4: LRPCA descriptor outline, adapted from [68]. In our method, we only use the green-boxed regions, two centered on each eye and one for the whole face.

subspaces are concatenated into a feature vector of size 192, that is further whitened by scaling each dimension to have a sample standard deviation of one on the training set.

2.3 Deep visual representations

Deep visual methods differs from feature engineered ones by introducing hierarchically learned representations, i.e., the ability to build complex concepts out of simpler ones, without depending completely on hand-crafted features. They are inspired by the biology of the mammal brain, organized in a deep architecture [3], where each level of abstraction corresponds to a different area of the cortex. The brain uses multiple hierarchical stages, specially in the visual system, to process perceptive information [75]. Fig. 2.5 shows how a deep learning model can build the concept of an image of a person, using simple concepts like edges and contours, learned directly from the raw pixels. Because of it, these deep visual representations are also called *data-driven representations*.

The architectures used in recent researches [17, 40, 48, 64] are composed of stacked layers, where each one of them receives information from the layer below, process it, and passes new stimulus to the layer directly above. The biologically-inspired intuition is that, as information flows through the network, each layer is able to come up with more complex concepts. Usually layers perform a sequence of operations: (a) linear filtering followed by nonlinear activation, modeling the simple cell behavior, (b) local pooling, modeling the complex cell behavior, and (c) local normalization, representing the competitive interactions among neurons. A number of parameters is necessary to determine the architecture of the network. They are called *hyperparameters* and are essential to achieve good performance. Some of them are: which filters should be used and in what order; the number of layers and how they are connected, etc.

Another important step in a deep architecture is the training procedure of the network, usually performed with back-propagation [72]. The technique determines the weights of the connections in the network by iteratively forward-propagating an input and comparing the actual output to the desired one. Trying to minimize the difference between the two outputs, this value is, then, propagated backwards and the weights are adjusted accordingly. Until the early 2000s, it was believed to be too difficult to train deep multi-layer neural networks [4]. Empirically, deep networks were outperformed by neural networks with a couple of hidden layers [84], possibly because gradient-based optimization initialized with random parameters got stuck near poor solutions [4]. However, in 2006, Hinton *et al.* [34] presented a greedy layer-wise unsupervised learning algorithm, using *restricted Boltzmann machines*, capable of speeding up the training of deep networks and achieving good results in handwritten digit recognition. Following the same line of research, other works [4, 37] explored variations of Hinton's method and its limitations.

Despite the advances in training, early 2000s face recognition datasets [10, 69, 57] were small and their images were captured in constrained conditions, with little variation in lightning, head pose, facial expression and occlusion. As a result of that, deep architectures still struggled with the lack of data, limiting their depth and number of parameters.

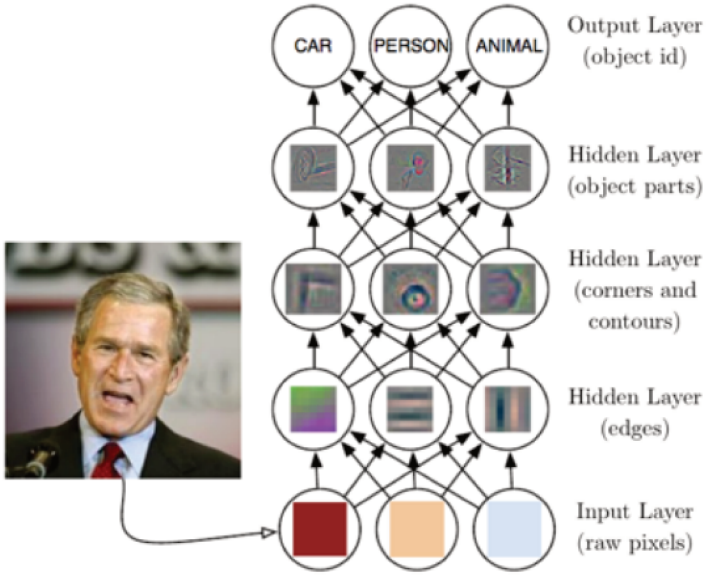


Figure 2.5: Illustration of a deep learning model adapted from [28]. Raw pixels values are given to the learning step through the input layer, or visual layer. Using the information from the previous layer, each hidden layer builds a slightly higher level abstraction, e.g., edges, local shapes, object parts, etc. This mapping is done up to the output layer, where the complex concept is retrieved to the user.

Chopra *et al.* [16] trained a six-layer *siamese architecture* [9], i.e., two identical convolutional networks with shared weights, to map a pair of face images to a low-dimensional space where a similarity metric could be used to verify a user's identity. Although siamese networks elegantly model the face verification task, they may not be suitable to mobile devices, since they require more computational effort than a single equivalent CNN.

The work of Cox and Pinto [18] achieved impressive results in the unconstrained *Labeled Faces in the Wild* dataset [39] with a deep architecture called L3+ representation. This representation can be seen as a three-layer CNN whose hyperparameters were found with brute-force optimization, while using random weights for the network's convolutional filters. Chiachia [13] improved the results by incorporating a supervised learning step on top of the original architecture. The author used a linear support vector machine (SVM) to learn person-specific characteristics from the output of L3+'s last layer.

As more data became available through the Internet, more complex deep architectures were designed, and data-driven methods have started to achieve lower error rates than engineered descriptors, since they are able to optimize features for the specific task at hand [28].

Krizhevsky *et al.* proposed the AlexNet architecture [48], a deep CNN that won the ImageNet Challenge (ILSVRC) [73] in 2012. As illustrated in Fig.2.6, it has stacked convolutional layers and fully-connected layers, adding up to a total of 60 million parameters. The authors also incorporated max-pooling and ReLU non-linearity after layers, while dropout [35] and data augmentation were used to reduce overfitting.

The work of Taigman *et al.* [82] proposed a face alignment method based on explicit 3D modeling of the face, as well as, DeepFace, a CNN with a slightly different architecture than AlexNet. It has three locally connected layers, i.e., layers that learn a set of filters

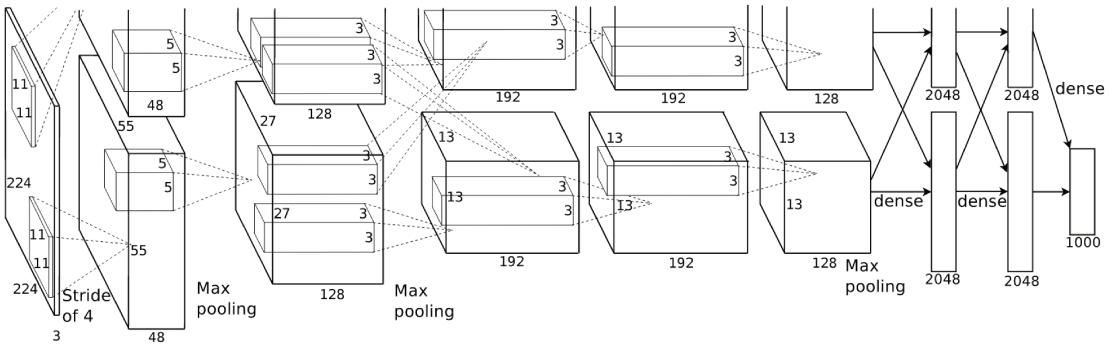


Figure 2.6: Representation of AlexNet architecture [48], with five convolutional layers — whose filters have size 11×11, 5×5 and 3×3 — and max-pooling operation following some of the them. At the end, three fully-connected layers feed a softmax of 1000 classes, representing the categories of 2012 ImageNet Challenge.

for each location in the feature map, instead of a shared filter for the whole map as a convolutional layer does. This allows the network to learn to respond according to different spatial positions, however, greatly increases the number of parameters; of its 120 million, locally and fully connected layers account for 95% of them.

FaceNet [74] uses a deep convolutional network to learn a direct embedding from the face images to a low-dimensional Euclidean space, where distances directly correspond to a measure of face similarity. The authors define loss function using triplets, i.e., a pair of face images from one identity (anchor and positive examples) and an image from another identity (negative example). The network is trained by minimizing the distance from the anchor to the positive representation, while maximizing the distance between the anchor and the negative. The authors point out that, although the method achieves highest accuracy up-to-date (99.63%) in the Labeled Faces in the Wild dataset [39], it requires huge amounts of data and computational resources, which may be an issue when running on a mobile device.

In [76], Simonyan and Zisserman evaluated the importance of increasing a network's depth to achieve good results and also how a composition of small convolutional filters (3×3) could have similar effect of a bigger one (5×5 or 7×7), while reducing the number of total parameters. They proposed VGGNet, a 16 or 19 layer CNN with 140 million parameters, that achieved state-of-the-art results in some tasks of 2014 ILSVRC [73]. A variation of VGGNet [64] was also trained for the face recognition task with 2.6 million pictures using a triplet-loss approach similar to [74]. This version, whose architecture is illustrated in Fig. 2.7, was used as basis for the architecture proposed in Section 4.3.

It was a fairly well-accepted idea that CNN layers should be stacked sequentially, however GoogLeNet [80] achieved outstanding results in ILSVRC 2014 Classification Challenge [73] by having layers processing the same input in parallel and then concatenating their outputs into a single one. This allows layers with distinct parameters to learn different characteristics and patterns from an input. The set of parallel layers, referred as Inception module (Fig. 2.8), is stacked sequentially to compose a 22 layers deep network. Although deeper, GoogleLeNet has 12x fewer parameters than AlexNet, since it does not use fully-connected layers — replaced by average pooling instead — and considers

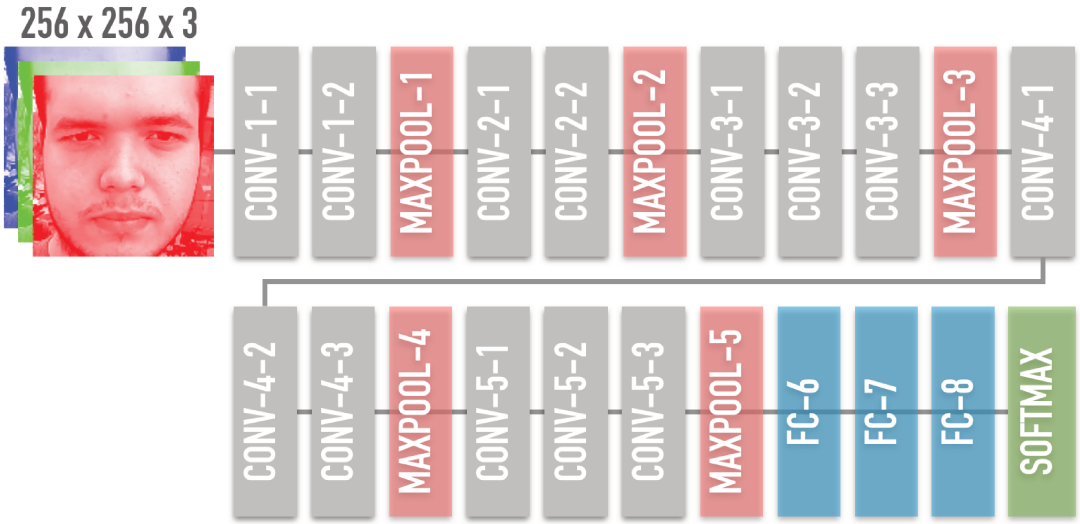


Figure 2.7: Representation of VGGNet architecture [64]. It receives a 256x256 RGB input image, has 13 convolutional layers (in grey) and 3 fully-connected layers (in blue). Detailed information about each layer, regarding its configuration, number of parameters and operations executed, is presented later in Table 4.1.

convolutions of size 1×1 to reduce the input map before 3×3 and 5×5 convolutions.

A standard convolution maps both spatial and cross-channel correlations at once, whereas Inception modules separate them by first learning cross-channel correlations using 1×1 convolutions and then mapping the spatial relation of their outputs via 3×3 and 5×5 convolutions. Besides improving accuracy, this also greatly reduce the number of required parameters. In [15], Chollet points out the similarity between the Inception module's architecture and *depthwise separable convolutions* — a spatial convolution performed independently over each channel of an input, followed by a 1×1 convolution, capturing the cross-channel correlations into a new channel space — and, using them, proposes *Xception* architecture, capable of outperforming GoogLeNet.

In addition to this type of convolutions, Xception also uses residual connections, a concept introduced by the ResNet [33]. In this work, He *et al.* manage to train very deep networks, with up to 1000 layers, by adding shortcut connections between some layers. These connections (illustrated in Fig. 2.9) force a layer to learn from a residual mapping, instead of the map from the previous layer, which improves model convergence and allows depth increase. In Chapter 5, we compare our proposed method to ResFace101 [58], a ResNet with 101 layers trained for the face identification scenario.

2.4 Mobile efforts

Even though the networks discussed so far achieve impressive results in their designated tasks, most of them are not suitable to run inside a mobile device. From the memory footprint viewpoint[1], AlexNet's model has 240MB [86] and VGGNet's has 580MB [89], while GoogLeNet has a smaller model of 50MB [88]. With this in mind, several approaches

[1]Comparison made with publicly available Caffe [44] models downloaded from Caffe Model Zoo [87].

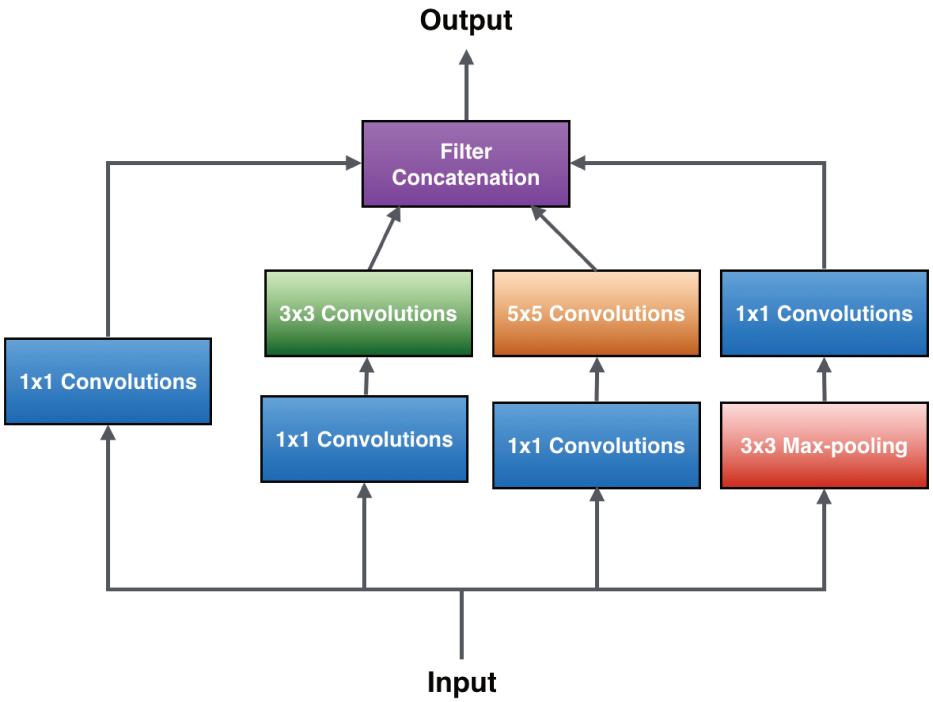


Figure 2.8: Inception module adapted from [80]. The 1×1 convolutions act as dimensionality reduction before the 3×3 and 5×5 convolutions, in order to decrease the number of input maps.

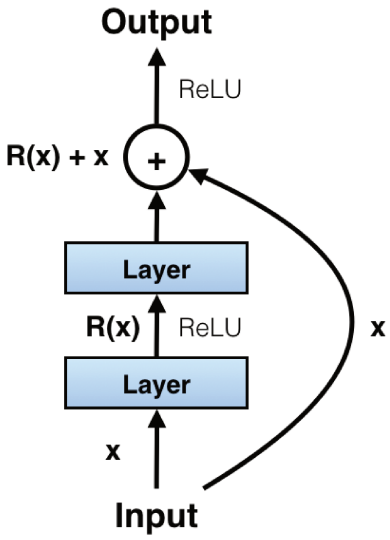


Figure 2.9: Shortcut connection skipping two layers, adapted from [33]. The layers in between are tasked to learn only *new information*, i.e., the residual $R(x)$ over the input x.

were designed to simplify, reduce and/or speed up existing architectures and techniques.

A fairly popular research line is referred as *model compression*, where a CNN model is compressed in a lossy process, decreasing model size while trying to maintain accuracy. Methods that fall in this line date to 1989, when LeCun *et al.* [49] proposed *Optimal Brain Damage*, an approach to remove weights from a neural network with the least impact in accuracy, determined by the second derivative of the objective function regarding each weight.

Inspired by [22], Denton *et al.* [23] argued that lower convolutional layers of deep CNNs are over-parametrized, i.e., they have a lot of redundancy that can be eliminated with almost no accuracy loss. Using single value decomposition they were able to transform weights matrices into a more computation and storage efficient representation.

Han *et al.* [31] proposed *Network Pruning*, replacing weights below a certain threshold with zeroes to form a sparse CNN, reducing the number of parameters of AlexNet by a factor of 9x and VGGNet by 13x, with no significant accuracy loss in both models. *Deep Compression* [30], an extension of the previous work, combined the last approach with quantization and Huffman encoding, decreasing the storage requirements of AlexNet by 35x and VGGNet by 49x.

Although model pruning and compression achieves interesting results, working with sparse CNN, quantization and encoding is not often supported by CNN libraries, such as Caffe [44], or may even require specialized hardware [30]. Considering this, Li *et al.* [50] proposed to prune filters instead of weights, indicating that it is a more structured way to prune, does not induce sparsity and is directly related to speed ups, since smaller feature maps means less matrix multiplications. For a given set of 2D filters of a convolutional layer, the selected filter to be pruned is the one that minimizes the sum of its absolute weights. This value gives an expectation of the magnitude of the output feature map, so a filter with a low sum contributes less to the overall performance of the network than the others from the same layer.

Other researches go in the line of designing compact and efficient network architectures, already tweaked to the limitations of low-powered devices. Bondi *et al.* [5] analyzed the resources used during the inner steps of the L3+ architecture [18] and were able to propose optimizations, bearing in mind the ratio between accuracy and consumed energy. The authors reduced by 94% the average energy consumption, while maintaining an accuracy rate of 86.73% with 1.06 seconds of processing time per image. Although in this research we used a deeper and more complex architecture than L3+, many of the insights from Bondi *et al.* were relevant in our designing process.

An architecture specially relevant to this work is *SqueezeNet* [40], which uses the concept of a repeatable block of layers from [80]. To constrain the number of parameters in the CNN, the authors propose to replace most of 3×3 filters with 1×1 — since the latter have 9x less parameters — and, in case of a layer with 3×3 filters, decrease the number of channels of the input map — which is directly related to the total parameters in that layer. Besides that, they argue that higher accuracy can be achieved by delaying downsampling (commonly performed with pooling or convolutions with stride > 1) to a late stage in the network. This last decision is also reinforced by work of He and Sun [32] that compared different CNN architectures and design choices under a time constraint. Considering these

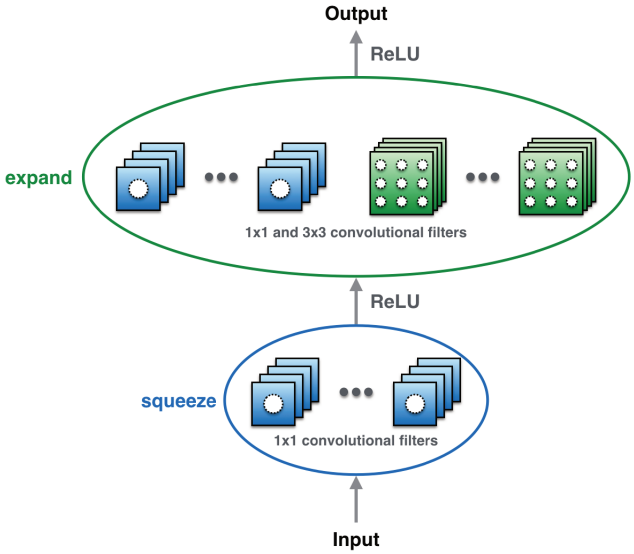


Figure 2.10: Fire module adapted from [40]. The number of 1×1 filters in the squeeze layer is always less than the total number of filters in the expand layer, in order to reduce the input map size to 3×3 convolutional filters.

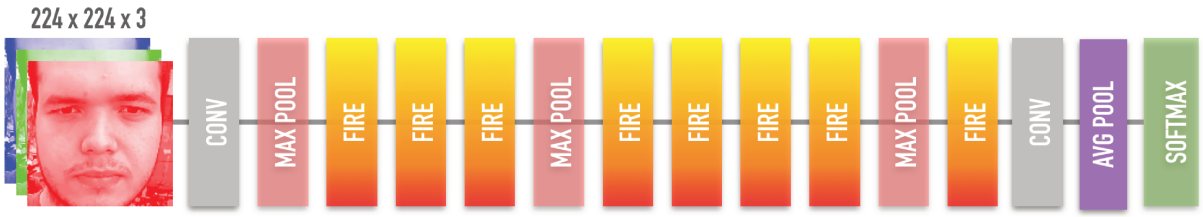


Figure 2.11: SqueezeNet architecture adapted from [40]. As we move forward in the network, the size of each Fire module (i.e., the number of filters in squeeze and expand layers) is gradually increased.

strategies, Iandola *et al.* conceived the *Fire* module, depicted in Fig. 2.10. It consists of two convolutional layers: a *squeeze* layer with 1×1 convolutional filters and a *expand* layer with both 1×1 and 3×3 convolutional filters. The squeeze layer, similar to the 1×1 convolutions of GoogLeNet's Inception module [80], acts as a bottleneck, reducing dimensionality before the expensive 3×3 convolutions, decreasing the number of input channels while also condensing relevant input information and discarding redundancy in it. Additionally to stacked Fire modules, SqueezeNet architecture (illustrated in Fig. 2.11) replaced traditional fully-connected layers with global average pooling. Fully-connected layers often correspond to a big percentage of a network parameters, are susceptible to overfitting and heavily dependent on dropout regularization [48, 35]. Whereas, average pooling has no parameters to be optimized — avoiding overfitting —, is more robust to spatial translation and its output feature maps can be interpreted as categories confidence maps [52]. With all this, SqueezeNet has a model 50x smaller than AlexNet (4.8MB), while, by complementing the design choices with Deep Compression [30], it is possible to achieve a model 510× smaller (0.47MB).

In [36], Howard *et al.* designed *MobileNet*, a flexible architecture with two special

hyperparameters α and ρ that limit the width and spatial resolution of each layer. While the former determines the number of output channels of each layer, by altering the number of convolutional filters; the latter limits the spatial resolution of the input images and, consequently, each layer's input. The authors analyze how these hyperparameters affect accuracy, number of parameters and performed operations, and they apply diverse network setups in different image recognition problems, comparing to state-of-the-art and pointing out the accuracy-performance trade-off. In the architecture proposed in our work, we apply a similar strategy of reducing the spatial resolution of each layer, by forwarding a smaller image to the network, in order to greatly decrease the number of operations performed by each layer.

Chapter 3

Datasets and Evaluation Protocol

Naturally, for a complex problem such as the one we tackl in this work, training data is pivotal for the success of the research. Training complex deep neural networks often requires huge quantity of data [74, 80, 82], representing a great number of identities in diverse capture conditions, such as illumination, hairstyle, occlusion, facial pose and expression. Although there are many publicly available datasets [39, 46, 68, 93], none of them is focused on selfie images. In this work, we have used four datasets: *RECOD Selfie Dataset*, *Motorola Selfie Dataset*, *Unicamp Video-Based Attack Database*, *Oulu-NPU database*. Examples of each dataset are presented in Fig. 3.1 and Fig. 3.2 and each image had its face detected and normalized as described in Section 2.1.

3.1 RECOD Selfie Dataset

The *RECOD Selfie Dataset* (RCD) is a public dataset[1], created during this research. It is formed by videos of 56 identities, filmed by themselves by pointing the frontal camera of a mobile device to their faces and recording videos of approximately 30 seconds. The videos were captured in outdoor and indoor environments, with different illumination conditions, as well as varying head pose and facial expression. The dataset was collected at University of Campinas (Unicamp), with the participation of members of its community. 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*. From these videos, we extract one frame per second, totalizing $2,873$ images, where most of them have 1080x1920 resolution, while a minority has 480x640.

3.2 Motorola Selfie Dataset

The *Motorola Selfie Dataset* (MOT) is a private dataset, created in cooperation with Motorola company. It consists of videos from 49 identities, captured in the same setup of RCD. From these videos, we extract one frame per second, totalizing $4,900$ images.

[1] dx.doi.org/10.6084/m9.figshare.5427142

(a) (b) (c)

Figure 3.1: Examples from **(a)** UVAD, **(b)** MOT and **(c)** OULU datasets.

3.3 Unicamp Video-Based Attack Database

The *Unicamp Video-Based Attack Database* (UVAD) [70] was specially developed to evaluate video-based face spoofing attacks, i.e., when an individual want to get access to a mobile device by presenting to the face recognition software a picture or a video of the device owner. It is comprised of valid access and attempted attack videos of 404 different people. Each user was recorded in two sections in different scenarios and lightning conditions. The database has 808 real access videos and 16,268 videos of attempted spoofing attacks, all in full high definition quality. Each real access video has around 9 seconds, from which we extracted 1 frame per second. Although UVAD is not focused on selfie images, individuals were recorded in a controlled frontal-facing view.

3.4 Oulu-NPU Database

The *Oulu-NPU Database* (OULU) [8] was created by the University of Oulu in Finland and the Northwestern Polytechnical University in China for face presentation attack detection in mobile scenarios. The dataset was used as part of the IJCB 2017 competition on generalized face presentation attack detection in mobile authentication scenarios [7].

It consists of 4,950 real access and attack videos, recorded from 55 individuals using the frontal camera of six mobile devices. The videos were captured in three sessions with different illumination conditions and background scene, and were split into three subsets

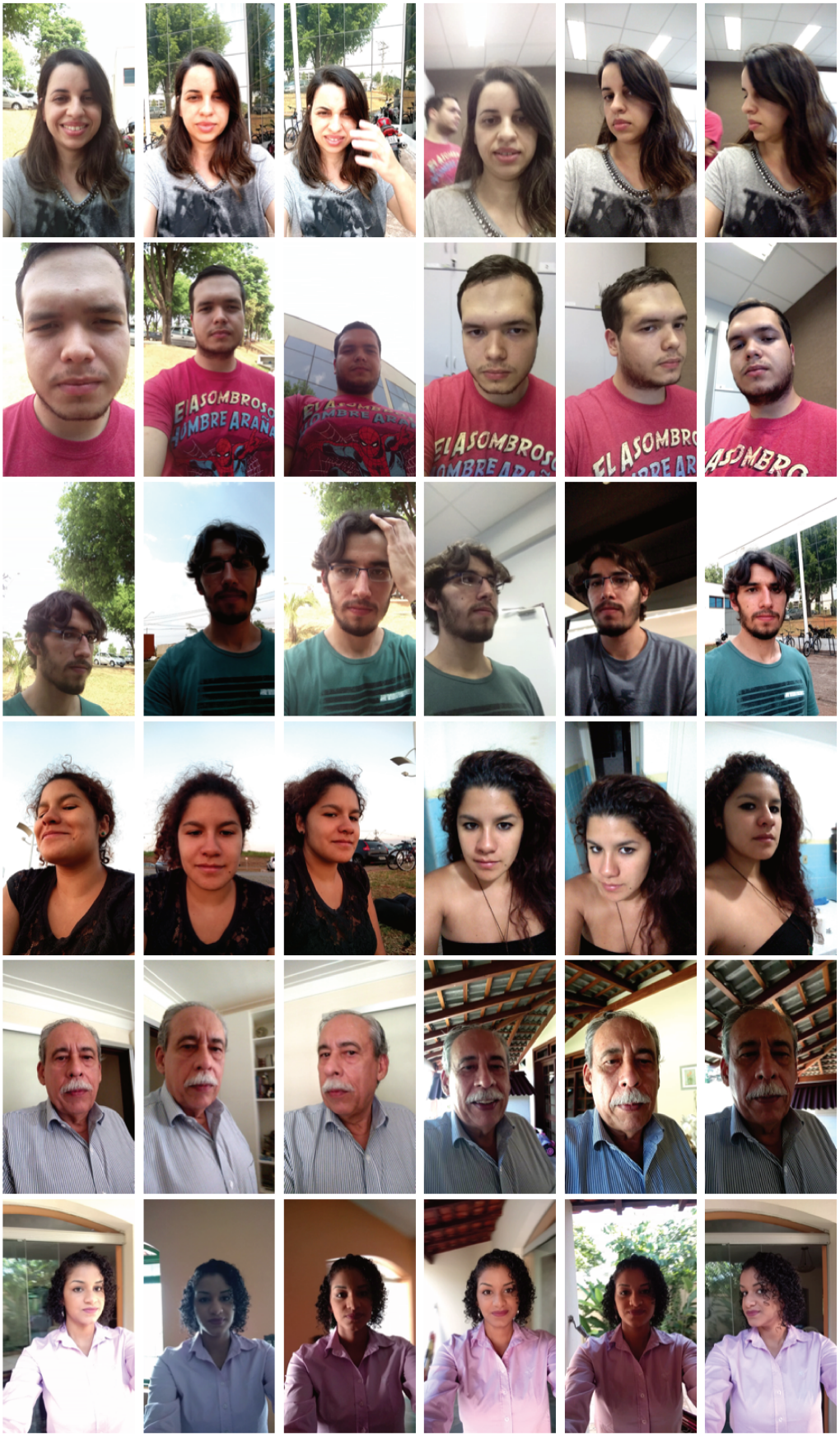


Figure 3.2: Examples from Recod Selfie Dataset. Each row corresponds to an identity, captured under different illumination conditions, head pose and/or facial expression.

for training, validation and test with no identity overlap.

In this research, we used the real access videos from OULU's Training and Development sets, totaling 35 identities and 630 videos, from which we extracted around 4 frames per second of each video.

3.5 Datasets summary

In this work, combining all datasets, we have worked with 564 identities, totaling 27,817 images in a wide range of illumination, background, hairstyle, facial pose and expression. Table 3.1 presents a summary of the four datasets used in this research.

Table 3.1: Datasets summary.

Dataset	Identities	Images	Pairs	Sessions per User	Capture Device
RCD	56	2,873	262,164	2	Smartphones
MOT	49	4,900	917,216	13	Smartphones
UVAD [70]	404	7,871	146,326	2	Digital Cameras
OULU [8][2]	35	12,173	4,222,188	3	Smartphones
Total	564	27,817	5,547,894		

As will be explained in Section 4.2, we have also built pairs with pictures pertaining to the same identity (*positive pairs*) and the same number of randomly selected pairs of images from distinct identities (*negative pairs*). We further organize the datasets into identity-disjoint train, validation and test sets. In order to verify generalization of the proposed method, the datasets were split in a way that allows cross-dataset experiments, as presented in Table 3.2.

Table 3.2: Train, validation and test sets.

Dataset	Set	Identities	Images	Pairs
MOT	Train	49	4,900	917,216
UVAD [70]	Train	404	7,871	146,326
OULU [8]	Train	20	6,965	2,419,628
RCD	Validation	14	575	34,382
RCD	Test	42	2,298	227,782
OULU [8]	Test	15	5,206	209,226 [3]

3.6 Evaluation protocol and metrics

Seven metrics that emphasize different aspects of a desirable solution were used for evaluation. We have specific objectives for three of them that we aim to achieve while designing our solution:

[2]We have only used Oulu-NPU's Training and Development (referred here as OULU-Test) sets.

[3]Due to time restrictions, only a randomly selected subset of pairs generated from OULU-Test set was used in our experiments.

- **True Positive Rate** (TPR) indicates how well a method is in authenticating the device owner. A true positive (TP) occurs when an owner's image is rightly classified. **We aim for a $\boldsymbol{TPR}$ above 90%**, which means the owner will have his or her access to the device wrongly denied once for each 10 attempts. This metric is given by Equation 3.1, where $|TP|$ is the number of true positives and N_{owner} is the number of verification attempts made with an owner's picture.

$$TPR = \frac{|TP|}{N_{owner}} \tag{3.1}$$

- **True Negative Rate** (TNR) indicates how well a method is in denying access to an intruder. A true negative (TN) happens when an intruder's picture is rightly classified as not being from the device owner. **We aim for a $\boldsymbol{TNR}$ above 99%**, which means an intruder would be allowed access once for each 100 attempts. This metric is calculated by Equation 3.2, where $|TN|$ is the number of true negatives and $N_{intruder}$ is the number of verification attempts made with an intruder's picture.

$$TNR = \frac{|TN|}{N_{intruder}} \tag{3.2}$$

- **Authentication time in mobile device** is also crucial to assess a good solution. For this metric evaluation, we used two smartphones: Motorola X Force, with 3GB RAM and Android 6.0.1 (**Smartphone A**) and Motorola Moto Z, with 3GB RAM and Android 7.1.1 (**Smartphone B**); and **aimed for our method to take around 1 second to authenticate or deny a face image on them**.

The following four metrics are used as comparative measures between two methods or CNN architectures, serving as secondary evaluation metrics in this work:

- **Accuracy** (ACC) indicates the overall performance of a method regarding true positives and true negatives. It is given by Equation 3.3, with $|TP|$ being the number of true positives, $|TN|$ being the number of true negatives and N is the number of verification attempts.

$$ACC = \frac{|TP| + |TN|}{N} \tag{3.3}$$

- **Number of multiplication and addition operations** in a CNN layer is related to its input and output sizes as well as the nature of the operation it performs. A comparison between the total amount of multiply-add operations of two architectures is a hardware-independent manner to estimate which one is faster.

- **Number of parameters in model** is related to the memory consumption during training and testing of a CNN. The more parameters it has, the bigger its model and the activation maps will be.

- **Model size in memory** is also important since mobile devices have limited memory space available.

Chapter 4

Methods

We propose a facial recognition method that consists of a two-tier solution tailored for the mobile environment, whose outline is illustrated by Fig. 4.1. Firstly, we use a set of user-specific classifiers tweaked to identify the owner's face pictures with a high confidence level. These classifiers are trained using two hand-crafted features — HOG and LRPCA — extracted from input face pictures. Since both features are fast to extract and the classifiers are trained with user-provided images, we seek high true positive rate without consuming too much time.

The second step consists of a group of classifiers trained to assess if a pair of faces belongs to the same identity or not. Each pair consists of the image being verified and one belonging to that user's gallery. We use two classifiers trained on HOG and LRPCA features separately and also a CNN trained on a combination of both images from each pair, that we called *hybrid image*. Based on the score and an acceptance threshold of each classifier, the user is authenticated or not.

We constantly check our phones for new messages and notifications [53], implicating that most authentication attempts are made by the device owner. Considering this, the proposed method must be fast and accurate for these frequent cases, however it is acceptable to take more processing time to deny an intruder in case he or she tries to authenticate.

We translated these ideas by having a fast 1st tier, whose confidence score is tested against two thresholds, one higher than the other. In case its score surpasses the highest threshold, the user is automatically authenticated; if the score is in between both thresholds, our method is not fully confident that the image belongs to the user, so we follow to the 2nd tier; whereas if the score is below the lowest threshold, access to the device is denied. In the 2nd tier, although more accurate, the fusion of hand-crafted and data-driven methods takes more time to process an input, so it is desirable to avoid this tier when possible.

Recent researches regarding CNNs have aimed to improve accuracy usually by making the network deeper and more complex [76, 79, 81]. However, this means bigger and more expensive models unfeasible to be used inside a smartphone. In this thesis, we propose a new CNN architecture, called *Hybrid Fire Convolutional Neural Network* (HFCNN), suitable for the mobile environment.

Also, as a way to better adapt the method to each user's unique face characteristics,

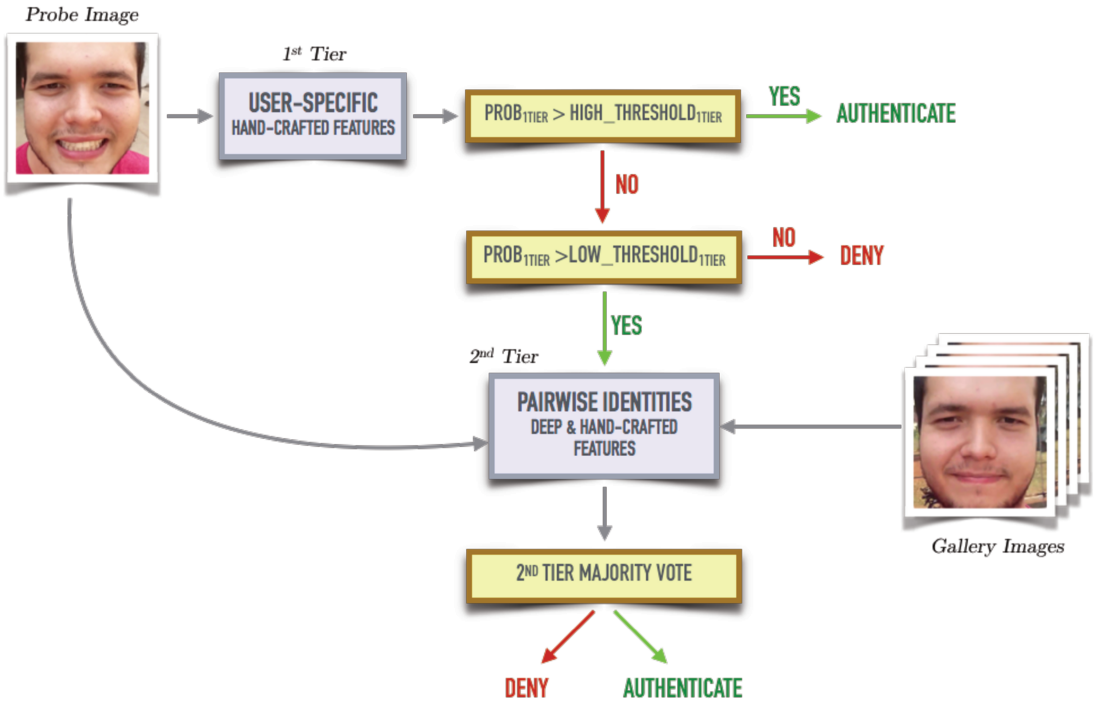


Figure 4.1: Outline of the 2-tiered solution proposed in this work. The 1st tier consists of a set of fast high confident user-specific classifiers, while the 2nd tier uses a fusion of deep and hand-crafted features to identify if a pair of images belongs to the same identity.

we propose a technique to automatically learn the acceptance threshold of the 2nd tier classifiers based on the face images provided by the user.

In the following sections we detail each component, discussing what decisions led to their design.

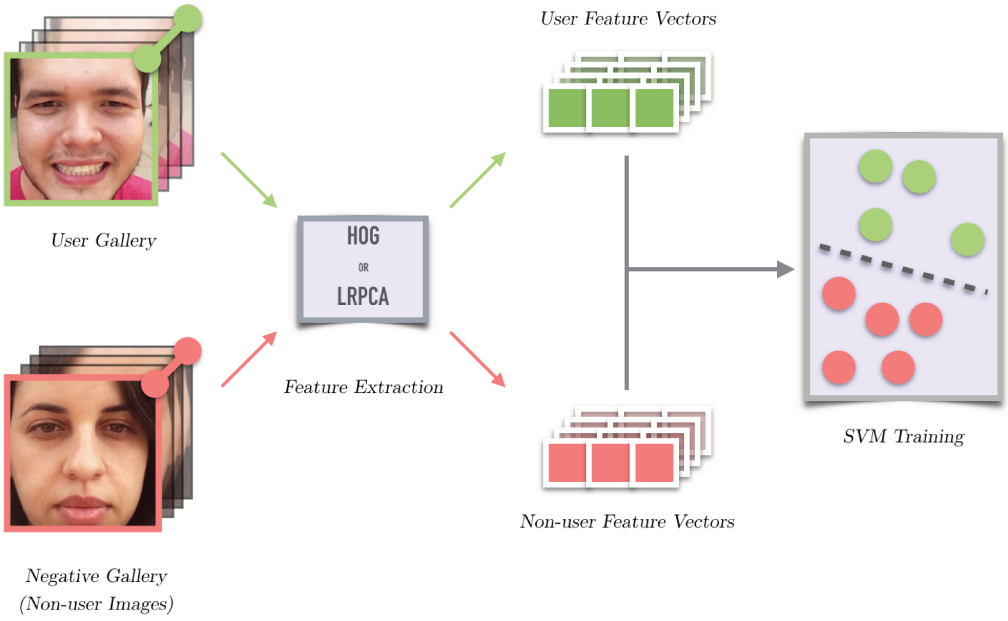
4.1 1st Tier: user-specific

As explored by Chiachia [13], it is possible to improve facial recognition by incorporating a supervised step to learn characteristics specific to a unique user. According to the authors, this is directly inspired by how the human brain is better at recognizing familiar faces than unfamiliar ones, probably because it leverages from previous experience to aid recognition.

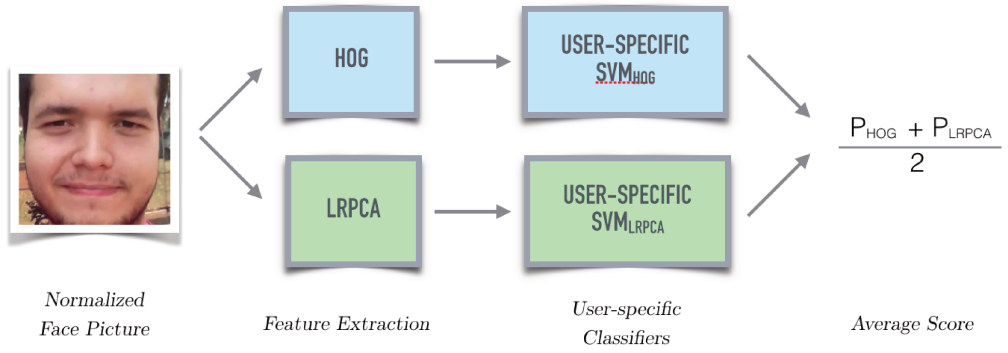
With fast advances in mobile technologies, smartphones, tablets and wearable devices have established their ubiquity in our society. It is expected that the number of mobile devices exceeds the world population by 2020 [60], achieving 1.5 device per capita. Mobile devices are mostly single user, which means we could leverage this proximity with the owner to optimize our solution to recognize his or her familiar face.

In order to do this, we train two linear SVM classifiers on top of HOG and LRPCA features separately. A gallery composed of user's images, captured during the enrollment, is used as *positive class*, while non-user images already embedded into the device beforehand are used as *negative class*. The training process is illustrated in Fig. 4.2a.

We have chosen these two hand-crafted features, because they are fast to compute and



(a) Training.



(b) Testing.

Figure 4.2: Pipeline for (a) training and (b) testing of the 1st tier step

have small representations, meaning they can be stored in memory without great impact.

This step's parameters are adjusted to achieve a high TPR and the training can be performed after the enrollment in an offline manner — e.g., when the device is idle — to decrease impact over usability.

During test (Fig. 4.2b), after being normalized, an input image has its features extracted and tested by the SVMs. The average of the classifiers confidence score — which is directly related to the distance between the feature vector and the SVM decision margin — is used as the final score for this tier.

4.2 2nd Tier: same-identity-or-not

The next step in the method seeks to identify if two images belong to the same individual. By slightly changing the target problem — from user-specific to image pair verification

— we wish to capture, using mostly the same features, different but complementary characteristics that together with the 1st tier improve overall recognition.

For this step, we have created pairs of images from the datasets. Each *positive* pair is composed by selecting two distinct images from the same identity, while a *negative* pair consists of two images from distinct identities. In order to have a balanced training, we randomly select among the negatives pairs — since their quantity is substantially greater — the same number of pairs as the positives. Within each pair, the first image is considered to be the *probe*, while the second is the *reference*. If instead of one reference we need to simulate a gallery with n images, then n photos are randomly selected among the ones depicting the reference's identity to compose the gallery tuple.

Since the training process of both hand-crafted and data-driven classifiers does not use device owner's pictures, it is done outside the mobile device. This allows us to use more complex and powerful models in this tier, that would not be possible if we resorted to in-device training.

4.2.1 Multiview hand-crafted classifiers

Once the enrollment is completed, the device owner will have a gallery of selfies in different *views*, i.e., similar photos with small variations in head pose, facial expression and illumination conditions. During the authentication we leverage from multiple views by comparing a probe to as many gallery images as possible, in what we refer as a *multiview comparison*. As the gallery increases in size and diversity, the method will have more information to authenticate or reject a new picture. Considering this, we build pairs of images, consisting of the probe and each image from the gallery.

To construct a feature vector F_{pair} for a pair of face images, feature vectors F_1 and F_2 are first extracted for each image of the pair. Then, the module of the difference and element-wise product of F_1 and F_2 are concatenated:

$$F_{pair} = [|F_1 - F_2|, F_1 \circ F_2]. \tag{4.1}$$

During training, considering a dataset of face images $D = \{I_1, \ldots, I_n\}$, we compute a set of pair feature vectors $S_{train} = \{F_{xy} | I_x, I_y \in D, x \neq y\}$. Note that if I_x and I_y depict the same person, then F_{xy} is labeled as *positive*, and *negative* otherwise.

The set S_{train} of pair feature vectors is used as input to a Logistic Regression classifier (LogReg), in order to learn a model able to make a prediction as to the probability of the input being *positive*.

The logistic regression is defined as

$$y = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n, \tag{4.2}$$

where $w_1, \ldots, w_n$ are the coefficients to be optimized, w_0 is the bias, and $x_1, \ldots, x_n$ are the components of one training sample F_{xy}. The output y is transformed into a probability using the logistic function:

$$p = 1/(1 + e^{-y}). \tag{4.3}$$

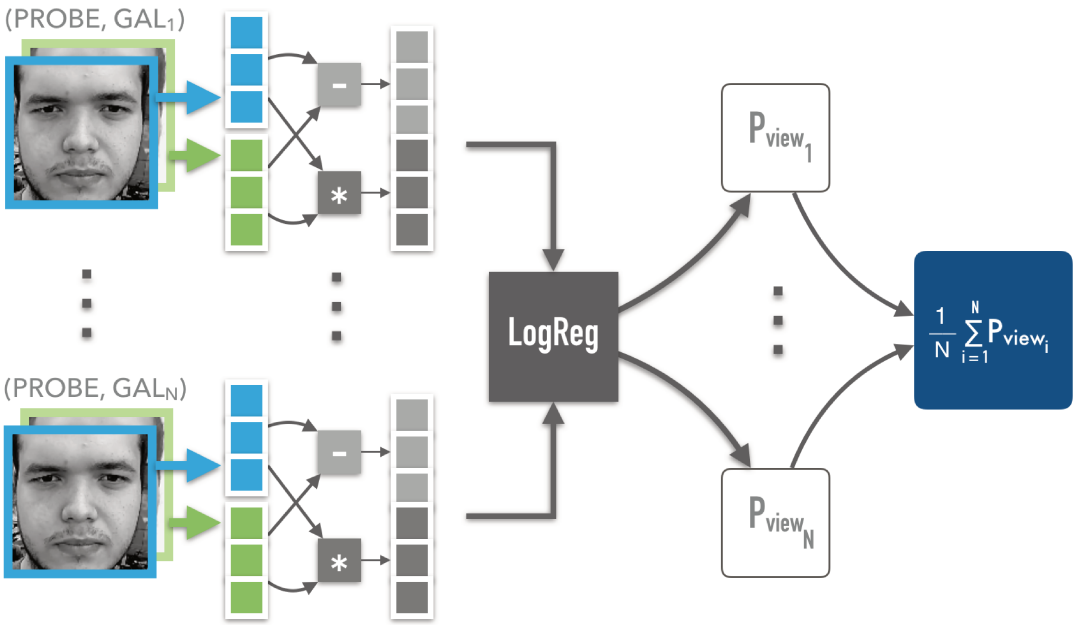


Figure 4.3: Outline of a multiview authentication, performed in the 2nd tier. Each pair consists of the probe and a gallery image that have their features extracted and combined into a feature vector. A logistic regression model outputs the probability of that pair belonging to the same identity and the average probability among all pairs will be combined with the data-driven output to determine the 2nd tier result.

We apply Stochastic Gradient Descent (SGD) to the problem of finding the coefficients for the logistic regression model. For each training instance F_{xy}, we calculate a prediction using the current values of the coefficients. The gradient of the function being optimized is computed, and the coefficients are updated. After training the model (finding the optimal coefficients), any test pair feature vector can be predicted as being *positive* or *negative*.

During test, considering a gallery of n face images $G = \{I_{gal_1}, \ldots, I_{gal_n}\}$, we compute a pair feature vector between a gallery image $I_{gal_i} \in G$ and the probe I_{probe} as:

$$F_{view_i} = [|F_{probe} - F_{gal_i}|, F_{probe} \circ F_{gal_i}] \quad , \quad i \in \{1, \ldots, n\}. \tag{4.4}$$

The set $S_{multiview} = \{F_{view_i}, \ldots, F_{view_n}\}$ of view feature vectors is used as input to the LogReg classifier, that outputs the probability p_{view_i} that the respective F_{view_i} represents a pair of images from the same identity. The final probability $p_{multiview}$ is the average of all p_{view_i}:

$$p_{multiview} = \frac{1}{n} \sum_{i=1}^{n} p_{view_i}. \tag{4.5}$$

We train a LogReg model on top of HOG features and another based on LRPCA features. In an authentication attempt (Fig. 4.3), a probe will yield a final probability for each type of feature — referred as p_{HOG} and p_{LRPCA} — that are combined with the output from the data-driven classifier to compose the 2nd tier final result.

4.2.2 Hybrid image and data-driven classifier

In literature, face recognition is usually approached as a multi-class problem, where a face image is assigned to one of many classes, each representing an identity. Thus, if we wanted to use a CNN for a pair of images, we would use the network as a feature extractor and feed it each image of the pair, construct the feature vector for the pair, and then train a classifier on top of it. The idea behind this is that, besides capturing patterns regarding each identity, the network's features also have information on how to separate general identities.

On the other hand, face verification is a binary problem, seeking to answer if two pictures depict the same person or not. A common data-driven approach in literature is to use CNNs with *siamese* networks [9], where each image of a pair is fed to the network that outputs feature vectors in a projection space where distances relate to identity similarity [36, 74, 80]. Besides learning the structural characteristics relative to faces, during training the network is optimized taking into account how the feature vectors are positioned in the projection space, i.e., trying to approximate vectors from the same identity and distancing the ones from different individuals.

However, in a resource-limited environment such as the mobile device, we want to limit the number of forward passes in a network without sacrificing accuracy. In this sense, we cannot use the multiview comparison from Section 4.2.1, since it implies multiple passes through the network.

Therefore, as a way to limit the amount of processing done by the network, but also to leverage from the user's gallery, we propose *hybrid images*. We combine in a single image the information regarding the probe and the gallery, as illustrated in Fig. 4.4. For a probe I_{probe} and a gallery of n face images $G = \{I_{gal_1}, \ldots, I_{gal_n}\}$, the input of the CNN is constructed as:

- Transform I_{probe} and each image of G to grayscale: I'_{probe} and $G' = \{I'_{gal_1}, \ldots, I'_{gal_n}\}$;
- Create the average image I'_{AVG} for the grayscale gallery G', where:

$$I'_{AVG} = \sum_{i=1}^{n} \frac{I'_{gal_i}}{n}. \tag{4.6}$$

- Create a single-channel image I_Z filled with zeroes with the same dimensions of I'_{probe} and I'_{AVG};
- Stack three channels $(I'_{probe}, I'_{AVG}, I_Z)$ to form the input.

Hybrid images are used as inputs to train the CNN whose architecture and training are discussed in Section 4.3. A softmax layer at the CNN's end outputs the probability p_{HFCNN} of a hybrid image representing the same identity.

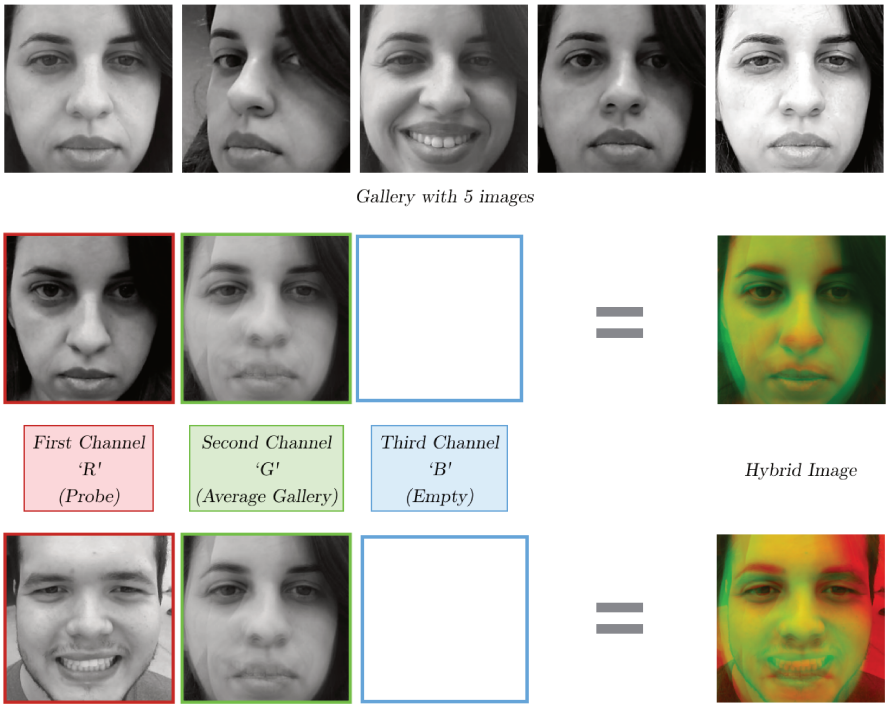


Figure 4.4: Generation of hybrid images, using a gallery with five images and two probes, one depicting the same identity as the gallery and another from a different person. The probe, the gallery average image and a *zero image* are stacked to compose the RGB hybrid image that is fed to the network. Our intuition is that the CNN is able to learn patterns that distinct same-identity hybrid images from different-identities ones.

4.2.3 2nd tier fusion and decision

Instead of a unique threshold for the 2nd tier, each classifier has a threshold associated with it. Since each method — HOG, LRPCA and HFCNN — captures different characteristics presented in data, we wanted to have different acceptance frontiers for each of them. Also, in Section 4.4, we propose a way to fine-tune each threshold based on the device owner's pictures. Therefore, given a classifier with threshold t and a pair with probability p, the classifier decision d is:

$$d = \begin{cases} \text{same person} & \text{if } p \geq t, \\ \text{not the same person,} & \text{otherwise.} \end{cases} \tag{4.7}$$

With three classifiers decisions — d_{HOG}, d_{LRPCA} and d_{HFCNN} — we use majority vote to determine if a probe is authenticated or not.

4.3 Hybrid-fire convolutional neural network

As discussed in Section 2.4, there has been an effort in literature to propose adequate CNN architectures for the mobile environment. Ultimately, we want a network with few parameters that perform as few operations as possible, while also capable of achieving great accuracy, TPR and TNR in our task.

Table 4.1: VGG-Face Architecture.

Layer	**Input Size**	**Filters (number / size)**	**Million Multiply-Add**	**Thousand Parameters**
conv1_1	3×224×224	64 / 3×3	86.7	1.73
conv1_2	64×224×224	64 / 3×3	1,850	36.86
conv2_1	64×112×112	128 / 3×3	924.84	73.73
conv2_2	128×112×112	128 / 3×3	1,850	147.46
conv3_1	128×56×56	256 / 3×3	924.84	294.91
conv3_2	256×56×56	256 / 3×3	1,850	589.82
conv3_3	256×56×56	256 / 3×3	1,850	589.82
conv4_1	256×28×28	512 / 3×3	924.84	1,180
conv4_2	512×28×28	512 / 3×3	1,850	2,360
conv4_3	512×28×28	512 / 3×3	1,850	2,360
conv5_1	512×14×14	512 / 3×3	462.42	2,360
conv5_2	512×14×14	512 / 3×3	462.42	2,360
conv5_3	512×14×14	512 / 3×3	462.42	2,360
fc6	512×7×7	4096[2]	102.76	102,760
fc7	4096×1×1	4096[2]	16.78	16,780
fc8	4096×1×1	2622[2]	10.74	10,740
		Total	15,478.76	144,994.33

Training a neural network from scratch requires a huge amount of images and time, therefore it is a common practice to fine-tune a model initialized with weights trained in a similar domain [95]. Since early layers learn low-level features, such as edges and color blobs, that usually are common to most image processing tasks, the fine-tune process can skip directly into optimizing the weights of deeper layers for more specialized concepts. This is also done by setting smaller learning rates for layers at the beginning of the network, that are increased at the final layers or the ones being trained from scratch.

This research's initial explorations with a data-driven method were done with VGGFace network [64], whose architecture is depicted in Fig. 2.7. Considering its impressive results in facial recognition, we used the model's weights as initialization for our architecture. As a starting point, we altered the last fully-connected layer to reflect the binary verification problem and we measured[1] the number of parameters and multiply-add operations for each convolutional and fully-connected layers. These statistics, exposed in Table 4.1, gave us information about the resource consumption of each layer and pointed us to possible layers that could be removed or altered in order to achieve a lighter network.

We opted to modify the layers at the network's end instead of the ones at the beginning, in order to preserve most of VGGFace initialization weights as possible. An alteration, for example, in the number of filters of conv4_1 layer would require to train from scratch all subsequent layers (conv4_2 to fc8).

[1]The number of multiply-add operations and parameters was measured using Netscope [29].

[2] Fully-connected layers are a special case of convolutional layers, where the size of the filters matches the size of the input data.

Table 4.2: Fire module hyperparameters for an image of size 224×224.

Layer	Type	Input Size	Filters (number/size)	Output Size	Thousand Parameters	Million Multiply-Add
squeeze1x1	Conv	512x14x14	64 / 1×1	64x14x14	32,768	6.42
relu_squeeze1x1	ReLU	64x14x14	-	64x14x14	-	-
expand1x1	Conv	64x14x14	256 / 1×1	256x14x14	16,384	3.21
relu_expand1x1	ReLU	256x14x14	-	256x14x14	-	-
expand3×3	Conv	64x14x14	256 / 3×3	256x14x14	147,456	28.90
relu_expand3×3	ReLU	256x14x14	-	256x14x14	-	-
concat	Concat	2x256x14x14	-	512x14x14	-	-
				Total	196,608	38.53

With this in mind, the first set of modifications was to remove fully-connected layers. Although responsible for only 0.8% of multiply-add operations, they account for 90% of the model's total parameters. Instead of fully-connected operations, recent architectures [52, 40] make use of global average pooling. Besides acting as a regularizer, which makes the network less prone to overfitting, it impels correspondence between previous convolutional layer's feature maps and each category.

As we move deeper in the architecture, the number of parameters in the convolutional layers increases, due to the increase in the number of filters being learned. Once the fc8, fc7 and fc6 are removed, all three conv5 layers account for 48% and 9% of the remaining parameters and multiply-add operations respectively. Besides that, they are responsible for learning most high-level concepts related to identities of VGGFace target domain, which are not appropriate to differentiate identities present in hybrid images. We removed conv5 layers replacing them with Fire modules from SqueezeNet [40]. The addition of these modules serves not only to compensate the huge quantity of removed parameters that would naturally decrease model accuracy, but also to add depth to the network without greatly increasing the parameter count. We present in Table 4.2 the Fire module hyperparameters used in our work, while Fig. 4.5 depicts how each layer is organized inside a module.

In order to reduce the number of operations performed by the network, we follow a similar strategy to MobileNets [36]. Rather than directly altering hyperparameters, we feed smaller hybrid images to the network. By reducing each dimension of the network's input image by half, the internal maps also shrink by same ratio, thus decreasing the amount of performed multiply-add operations. When the HFCNN is fed with a 224×224 image, the Fire's internal activation maps are of size 14x14, decreasing to 7×7 when a 112×112 input image is used. Consequently, the number of multiply-add operations falls from 38.53 million to 9.63 million (Table 4.4).

A simplified outline of Hybrid-Fire Convolutional Neural Network architecture is illustrated in Fig. 4.6, whereas Tables 4.3 and 4.4 expose respectively each layer setup and number of multiply-add operations for inputs of size 224×224 and 112×112.

Besides the proposed alterations, other details related to the architecture, training and testing procedure, and deep learning framework also are fundamental in our approach:

- Dropout [77] with rate of 50% was applied on *fire6, fire7* and *fire8* layers for better regularization;

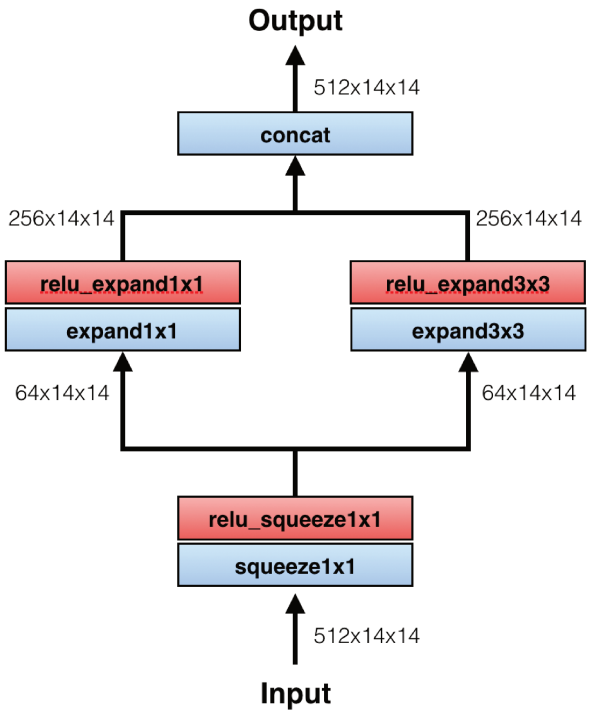


Figure 4.5: Layer organization inside a Fire module. The internal maps dimensions are 14x14 for an input image of size 224×224, and 7×7 for a 112×112 input. The *concat* layer receives two maps with 256 channels and outputs the concatenation of both maps in the channel axis.

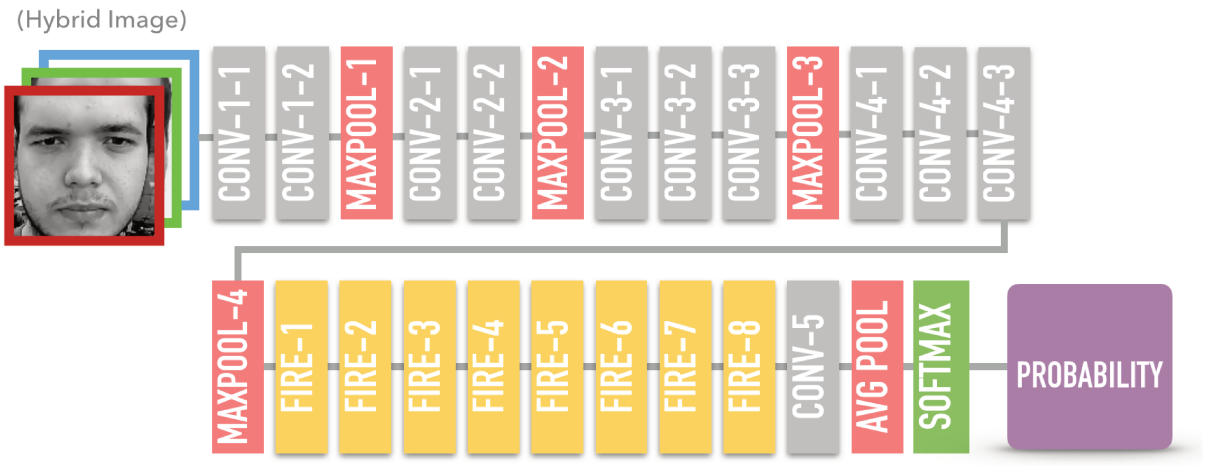


Figure 4.6: HFCNN architecture outline. Each convolutional layer (gray boxes) are followed by ReLU operations and each Fire module (yellow boxes) have the internal configuration depicted by Fig. 4.5.

Table 4.3: HFCNN architecture, with correspondent internal maps for input images of size 224×224 and 112×112.

		Image Size			
		224×224		**112×112**	
Layer	**Filters (number/size/stride/padding)**	**Input Size**	**Output Size**	**Input Size**	**Output Size**
conv1_1	64 / 3×3 / - / 1	3x224×224	64x224×224	3x112×112	64x112×112
relu1_1	-	64x224×224	64x224×224	64x112×112	64x112×112
conv1_2	64 / 3×3 / - / 1				
relu1_2	-				
maxpool1	- / 2x2 / 2 / -	64x224×224	64x112×112	64x112×112	64x56x56
conv2_1	128 / 3×3 / - / 1	64x112×112	128x112×112	64x56x56	128x56x56
relu2_1	-	128x112×112	128x112×112	128x56x56	128x56x56
conv2_2	128 / 3×3 / - / 1				
relu2_2	-				
maxpool2	- / 2x2 / 2 / -	128x112×112	128x56x56	128x56x56	128x28x28
conv3_1	256 / 3×3 / - / 1	128x56x56	256x56x56	128x28x28	256x28x28
relu3_1	-	256x56x56	256x56x56	256x28x28	256x28x28
conv3_2	256 / 3×3 / - / 1				
relu3_2	-				
conv3_3	256 / 3×3 / - / 1				
relu3_3	-				
maxpool3	- / 2x2 / 2 / -	256x56x56	256x28x28	256x28x28	256x14x14
conv4_1	512 / 3×3 / - / 1	256x28x28	512x28x28	256x14x14	512x14x14
relu4_1	-	512x28x28	512x28x28	512x14x14	512x14x14
conv4_2	512 / 3×3 / - / 1				
relu4_2	-				
conv4_3	512 / 3×3 / - / 1				
relu4_3	-				
maxpool4	- / 2x2 / 2 / -	512x28x28	512x14x14	512x14x14	512x7x7
fire1	-	512x14x14	512x14x14	512x7x7	512x7x7
fire2	-				
fire3	-				
fire4	-				
fire5	-				
fire6	-				
fire7	-				
fire8	-				
conv5	2 / 1×1 / - / -	512x14x14	2x14x14	512x7x7	2x7x7
relu_conv5	-	2x14x14	2x14x14	2x7x7	2x7x7
globalAVG_pool	-	2x14x14	2x1x1	2x7x7	2x1x1
softmax	-	2x1x1	2x1x1	2x1x1	2x1x1

Table 4.4: HFCNN parameters and operations for input images of size 224×224 and 112×112.

		Million Multiply-Add	
Layer	**Thousand Parameters**	**224×224**	**112×112**
conv1_1	1.73	86.7	21.68
conv1_2	36.86	1,850	464.42
conv2_1	73.73	924.84	231.21
conv2_2	147.46	1,850	462.42
conv3_1	294.91	924.84	231.21
conv3_2	589.82	1,850	462.42
conv3_3	589.82	1,850	462.42
conv4_1	1,180	924.84	231.21
conv4_2	2,360	1,850	462.42
conv4_3	2,360	1,850	462.42
fire1	196.63	38.53	9.63
fire2			
fire3			
fire4			
fire5			
fire6			
fire7			
fire8			
conv5	1.02	0.2	0.05
Total	9,208.23	14,270	3,570

- During training, layers from *conv1* up to *conv4* were initialized with VGGFace weights, whereas *Xavier* initialization [27] was applied for Fire modules and *conv5*. By inspecting the size of the network input, Xavier produces initial weights that guarantee a signal will remain in a reasonable range of values while being forwarded in the network;

- *Adam* [47] method was used to optimize the network. Similar to stochastic gradient descent (SGD), Adam is a gradient-based optimization method that computes adaptive learning rates for each parameter from estimates of first and second moments of the gradients. As suggested by the authors, we use $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$ for its internal parameters that approximate moments.

- We used mini-batches of 256 hybrid images balanced for both classes, shuffling the training set after each epoch.

- Although the use of pair of images instead of single ones greatly increases the amount of training pictures, we also used data augmentation to further improve the method's robustness to small variations. After normalization, both images of a pair are reshaped to 256x256 or 128x128 — depending on desired network configuration — before hybrid construction. The following augmentations are applied to the resultant hybrid image:

  - **Crop:** during training, each time the optimization visits a hybrid image, a random position crop of 224×224 (for 256×256 input images) or 112×112 (for 128×128 input images) is performed and fed to the network. However, during test, the crop is always done in the center of the image.
  - **Mirror:** each training image has a 0.5 probability of being horizontally flipped before being processed by HFCNN.

- Loss in training and validation sets was registered during optimization. Training was performed until convergence in validation set or validation loss started to increase (*early stopping*).

- The deep learning framework Caffe [44] was used for network definition, training and experiments. It was selected for its simplicity, the numerous available pre-trained architectures [87] and for having a stable version ported for Android platform [71];

- All training and experiments were performed in a NVIDIA GeForce TITAN X (Pascal) GPU with 12GB of memory.

4.4 User-specific threshold learning

Defining a probability threshold above which a probe is authenticated is not a trivial task. Usually choosing a threshold requires balancing the trade-off between TPR and TNR; a lower threshold means most attempts will be authenticated, increasing TPR at the cost of lowering TNR, allowing access to intruders. On the other hand, a solution

with a higher threshold is more rigorous on which images will be authenticated, requiring a higher confidence that the probe depicts the device owner. Consequently, this reflects in a TPR decrease — some user's images will be mistakenly negated — although it reduces the probability an intruder will gain access to the device, i.e., higher TNR.

All methods implemented in the 2nd tier of our solution analyze pairs of face pictures and determine if they belong to the same identity and, in contrast to the 1st tier, this is done without using any information related to the device owner identity. Instead of a unique threshold for the average of the three probabilities p_{HOG}, p_{LRPCA} and p_{HFCNN}, we define a particular threshold associated to each method, in a way to better adjust how strict or tolerant the final solution is to their individual decisions.

Given the wide range of facial attributes, there might be pairs of identities that are easier to differentiate and, not only that, as complementary techniques with intrinsic weak and strong points, a pair may be classified with more confidence by a method than by other. Additionally, with a limited-sized training, the proposed solution may be deployed to a device whose owner's face has completely different characteristics than those present in training, in which case, a pre-determined set of thresholds may hinder the method's performance.

Considering this, we propose a flexible technique to automatically choose the acceptance threshold of the 2nd tier classifiers using images from the gallery, balancing both desired TPR and TNR. This *user-specific threshold learning* can be performed inside the mobile device in an offline manner, i.e., when the device is idle, in order to avoid impact in user experience.

Given the user's gallery of n face images $G = \{I_{gal_1}, \ldots, I_{gal_n}\}$ and a negative gallery of m face images belonging to different identities $O = \{I_{neg_1}, \ldots, I_{neg_m}\}$:

- For each $I_{gal_i} \in G$, construct m sets $S_{i,j}$ using l randomly sampled images from $G - \{I_{gal_i}\}$, with $l < n$, $i \in [1, n]$ and $j \in [1, m]$;
- The positive set P consists of the tuples:

$$P = \{(I_{gal_1}, S_{1,1}), \ldots, (I_{gal_1}, S_{1,m}), \ldots, (I_{gal_n}, S_{n,m})\}; \tag{4.8}$$

- Construct n sets S_i, where S_i consists of all images from $G - \{I_{gal_i}\}$, for $i \in [1, n]$;
- The negative set N consists of the tuples:

$$N = \{(I_{neg_1}, S_1), \ldots, (I_{neg_1}, S_n), \ldots, (I_{neg_n}, S_n)\}; \tag{4.9}$$

- Run each tuple of P and N for each 2nd tier method (HOG, LRPCA and HFCNN) and register their probability;
- For each method, perform a line search for the threshold, maximizing a desired metric (e.g. accuracy or *F-score*).

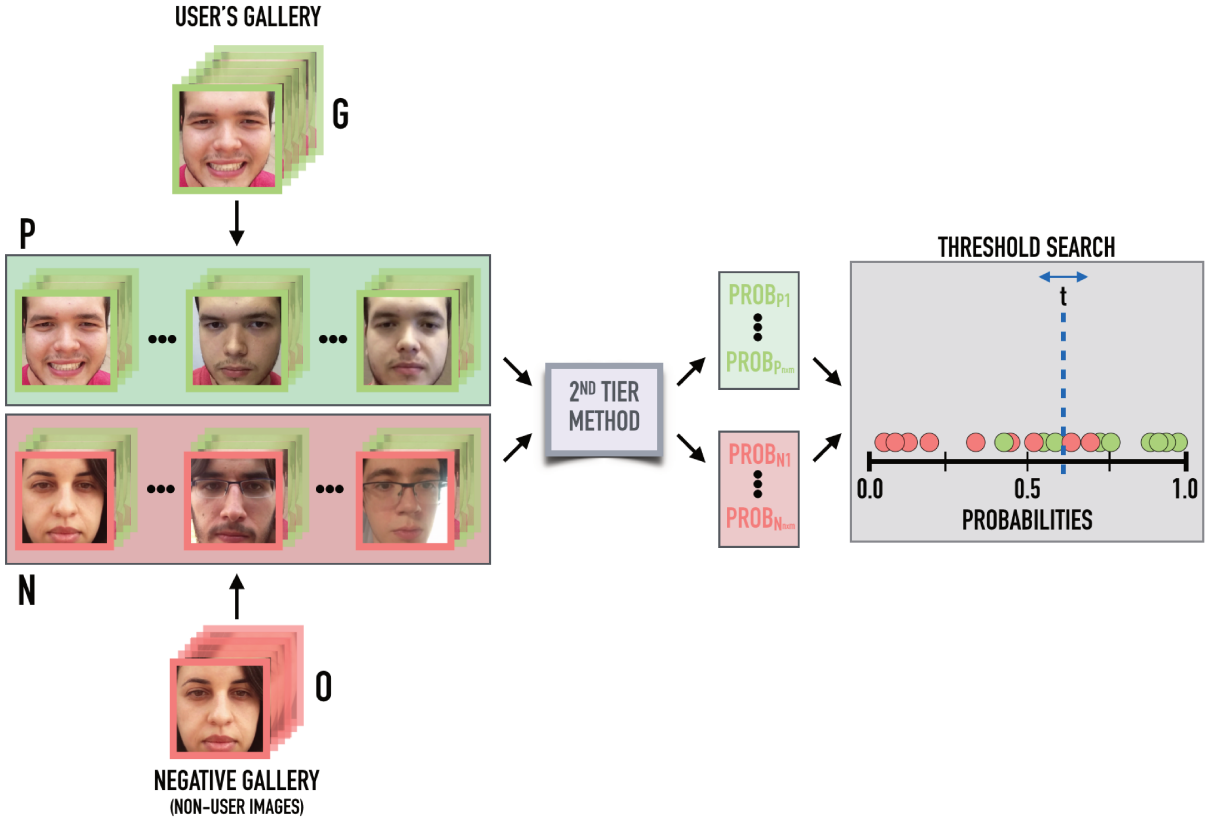


Figure 4.7: User-specific threshold learning outline.

- In case of a draw, where multiple thresholds maximize the desired metric, the highest threshold is selected in order to increase TNR.

In this work, during threshold search, we maximized accuracy, while trying to maintain $TPR > 0.9$ and $TNR > 0.99$. Additionally, in this research, we have used galleries of $n = m = 10$ images, which gave us equal-sized positive and negative sets of 100 tuples. For the positive tuples, we sampled $l = 7$ images from the user's gallery G. Fig. 4.7 illustrate the pipeline for a single 2^{nd} tier method.

Since it requires all tuples to be processed by each of this tier's classifiers, the size of both sets pose a challenge to this method, specially for slower methods such as CNNs. Suppose each forward takes 1 second, it would need around 3.5 minutes to process 200 tuples. Because of that, we suggest this step to be done when the device is idle to reduce the usuability impact.

Chapter 5

Results

The complete solution designed in this research is the result of the combination of a series of smaller, simpler and complementary techniques. In this chapter, we assess the impact of these individual components, not only to evaluate the whole system's performance, but also as a way to examine the possibility of integrating them into other methods or applying them to other problems.

For most experiments discussed in this chapter, MOT, UVAD and OULU-Train datasets were used as training sets, while RCD-Test and OULU-Test were test sets (Table 3.2). We also used RCD-Validation to select the best models when fine-tuning CNNs. The only experiments that had different data setup were related to the user-specific threshold learning (Section 5.5). Besides that, within each set, identity pairs and galleries were constructed following the steps from Section 4.2.

As face detection was not the focus of this work, for experimental purposes, we use a popular online API[1] or a C++ library[2], which returns, for a face image, the location of each eye center. This information is subsequently used for the normalization purposes exposed in Chapter 2. For the mobile implementation, it is crucial to use a fast local method to detect faces and, for this reason, we opted for using the already available Android APIs[3].

5.1 Multiview and hybrid images

The multiview approach leverages from the existence of a user's gallery to improve verification. By comparing a probe to all gallery images and averaging the individual results, it is possible to achieve a better accuracy at the cost of an increased processing time.

In Table 5.1 we explore different multiview setups, varying feature type and number of gallery images used. For these experiments, we extracted features from each image of the pair using HOG, LRPCA or VGGFace's *fc8* layer and constructed the final feature vector by concatenating the absolute difference with the element-wise multiplication of the individual feature vectors, as explained in Section 4.2.1. We trained a Logistic Regression classifier with python library scikit-learn [66] on the feature vectors from training pairs,

[1] `http://www.faceplusplus.com/demo-detect/`, as of January 2017.
[2] `http://dlib.net/`, as of June 2017.
[3] `http://developers.google.com/vision`, as of August 2017.

Table 5.1: Multiview exploration for HOG, LRPCA and VGGFace. Accuracy, TPR and TNR increase as more gallery images are used for authentication.

		RCD-Test			OULU-Test		
Feature	**Size of Gallery**	**Accuracy (%)**	**TPR (%)**	**TNR (%)**	**Accuracy (%)**	**TPR (%)**	**TNR (%)**
HOG	-	81.60	74.30	88.90	81.85	77.27	86.42
	3	87.21	81.55	92.88	85.08	80.59	89.57
	5	88.70	83.60	93.79	85.43	81.09	89.76
	7	89.47	84.73	94.21	85.63	81.24	90.02
	10	**90.05**	**85.58**	**94.52**	**85.83**	**81.50**	**90.15**
LRPCA	-	83.05	85.79	80.31	89.23	96.01	82.45
	3	89.14	92.37	85.90	92.57	99.00	86.14
	5	90.71	94.01	87.40	93.37	99.60	87.14
	7	91.50	94.73	88.28	93.70	99.86	87.54
	10	**92.19**	**95.37**	**89.00**	**93.92**	**99.96**	**87.89**
VGGFace	-	96.40	94.57	98.23	88.90	83.19	94.61
	3	98.07	97.25	98.89	90.06	85.10	95.02
	5	98.46	97.92	99.00	90.51	85.90	95.12
	7	98.56	98.08	99.05	91.01	86.83	95.18
	10	**98.69**	**98.29**	**99.09**	**91.57**	**87.95**	**95.20**

using 10-fold cross-validation to find the best hyperparameter setup. Finally, each testing pair had its feature vector constructed and classified.

For all three methods, the use of multiple images during testing significantly improves the performance. However, it also means that for each authentication attempt we will process a pair for each gallery image used in multiview. This may not be an issue with HOG and LRPCA since they are considerably faster (in the order of milliseconds), however processing multiple pairs with data-driven methods may not be possible in a real-time application. To avoid this, instead of storing the gallery images, it is preferable to store their extracted features and, when authenticating, only the probe is processed before building the multiview feature vectors. However, this may still pose an obstacle in applications where the gallery needs to be frequently updated.

Considering this, hybrid images offer an advantage and a disadvantage in relation to multiview. They directly halve the number of forward passes of a network by combining probe and gallery images into one, saving a lot of processing time for a gallery with several images. Nevertheless, since they encode pair information, it is not possible to store the features related to the gallery beforehand and only process the probe at testing time. This last point has a negative impact on multiview's efficiency and, in order to circumvent this, we proposed the hybrid image setup from Section 4.2.2, using the probe and the average image of the whole gallery. In our experiments, we have explored several hybrid image formulations, obtaining the most promising results with two of them:

- **(Probe, Gallery Image, 0):** we stack the probe as the first channel, a gallery image as the second and an image where every pixel has zero value as the third.

Table 5.2: Hybrid-image exploration for fine-tuned VGGFace and both versions of HFCNN. The hybrid formulation built with the gallery average image achieves a better result in most cases while also decreasing the number of forward passes in the networks.

			RCD-Test			OULU-Test		
Architecture	**Hybrid Formulation**	**Size of Gallery**	**Accuracy (%)**	**TPR (%)**	**TNR (%)**	**Accuracy (%)**	**TPR (%)**	**TNR (%)**
VGGFace Fine-tuned	(Probe, Gal, 0)	1	84.37	73.22	95.52	86.01	73.88	98.14
		10	91.53	83.89	99.17	92.61	85.51	99.71
	(Probe, Avg Gal, 0)	10	**92.80**	87.66	97.94	**94.83**	91.74	97.93
HFCNN 224×224	(Probe, Gal, 0)	1	88.22	87.58	88.87	87.38	84.46	90.29
		10	**96.08**	95.73	96.42	95.81	97.43	94.20
	(Probe, Avg Gal, 0)	10	93.82	90.89	96.76	**97.69**	96.72	98.65
HFCNN 112×112	(Probe, Gal, 0)	1	85.23	79.53	90.93	86.23	82.31	90.14
		10	93.53	90.31	96.76	91.38	88.32	94.44
	(Probe, Avg Gal, 0)	10	**93.67**	89.22	98.11	**94.79**	92.54	97.05

Since this formulation can be used with the multiview approach, besides reporting results for a single image, we also experiment with a gallery of 10 images.

- **(Probe, Gallery Average Image, 0):** as a way to avoid processing multiview's multiple pairs, we build the hybrid image by averaging 10 gallery images. This formulation is the same as the one explained in Section 4.2.2.

We have fine-tuned VGGFace and HFCNN architectures with these types of hybrid images, using the same setups of dropout, data augmentation and optimization algorithm explained in Section 4.3. We experimented with HFCNN with both 224×224 and 112×112 input sizes, however HFCNN 112×112 versions had its weights pre-initialized from the correspondent HFCNN 224×224 model before fine-tuning. We tested the pairs from RCD-Test and OULU-Test datasets and present the results in Table 5.2.

These experiments reinforce the importance of comparing the probe to more than one gallery image. Similar to the multiview experiments, the accuracy is improved as more images are used, either by averaging the scores of 10 hybrid images as well as processing a single hybrid image built from the average image of the gallery. This is due mostly because when multiple images are taken into account the influence of intra-class variations — such as illumination, occlusion, imperfections on face alignment, make-up or facial expression — are lessened, allowing the method to focus on the characteristics that differentiate two identities.

Both hybrid formulations have similar performances, with the second setup achieving a slightly better accuracy in most cases. More importantly, since it average all gallery images, this formulation allows a single network forward pass despite the gallery's size, which is important to reduce processing time.

5.2 Fusion of hand-crafted and data-driven features for the 2nd Tier

In Table 5.3, we explore different combinations of the methods employed in the 2nd tier in order to assess their overall importance to the solution. The fusion result is the majority vote of the decision of each individual method (using a default threshold of 0.5) that a probe belongs to the identity depicted in the gallery. For efficiency purposes, we have selected HFCNNs versions trained with hybrid images constructed with the average of a gallery of 10 images, while multiview approach with the same amount of pictures was used for HOG and LRPCA. We expose in Table 5.4 a time estimate to process a single image for each individual method.

Table 5.3: Fusion of hand-crafted and data-driven methods. As each feature capture different facial attributes, their fusion achieves better results than each one separately.

	RCD-Test			**OULU-Test**		
Method	**Accuracy (%)**	**TPR (%)**	**TNR (%)**	**Accuracy (%)**	**TPR (%)**	**TNR (%)**
HOG **(I)**	90.05	85.58	94.52	85.83	81.50	90.15
LRPCA **(II)**	92.19	95.37	89.00	93.92	99.96	87.89
HFCNN 224×224 **(III)**	93.82	90.89	96.76	97.69	96.72	98.65
HFCNN 112×112 **(IV)**	93.67	89.22	98.11	94.79	92.54	97.05
I + II	94.44	92.52	96.36	95.84	97.81	93.88
I + III	94.69	91.25	98.13	97.83	96.74	98.93
II + III	94.73	92.07	97.38	**98.37**	97.91	98.84
I + II + III	**95.49**	92.55	98.43	98.28	97.69	98.87
I + IV	94.23	89.78	98.67	94.80	92.41	97.20
II + IV	94.30	90.32	98.27	95.32	93.47	97.17
I + II + IV	**95.12**	91.44	98.80	**95.38**	93.52	97.24

Table 5.4: Time analysis for 2nd tier methods for a single image. HOG and LRPCA feature extraction have the advantage of being very fast, allowing them to process multiple images per authentication attempt without impacting device usability. Whereas, HFCNN offers an overall better accuracy, at the cost of a longer processing time.

	Processing Time (ms)	
Method	**Smartphone A**	**Smartphone B**
HOG	0.34	0.44
LRPCA	0.67	0.98
HFCNN 224×224	4,567.00	2,429.10
HFCNN 112×112	1,153.80	798.40

Besides outperforming pairwise combinations, the fusion of HOG, LRPCA and HFCNN almost achieves the aimed TPR and TNR (90% and 99%, respectively). Regarding both HFCNN architectures, it is important to notice, by concomitantly examining Table 5.3

and 5.4, the trade-off between a better accuracy (fusion with HFCNN 224×224) and a smaller processing time (combining with HFCNN 112×112). However, both architectures are considerably slower when compared with HOG and LRPCA.

In this sense, to further decrease the processing time of an authentication, it was necessary to either speed up HFCNN or to limit the frequency an image would be processed by the neural network. In view of the difficulty to make HFCNN faster without compromising accuracy, this motivated us to introduce the user-specific tier, as a preliminary step to filter the most common authentication cases in practice — i.e., the owner trying to access his or her device — using only hand-crafted features for efficiency.

5.3 User-specific verification for the 1st tier

Training or fine-tuning a CNN inside a mobile device is a costly and time-consuming task, besides strongly dependent on the framework or library being used and, because of that, it is preferable to embed an already-trained CNN to the mobile instead. However this also implicates that it is not straightforward to adapt the model with specific facial features of the device owner. Differently, HOG and LRPCA can be used to train a classifier inside the device, since they are features that are quickly extracted and have low memory footprint.

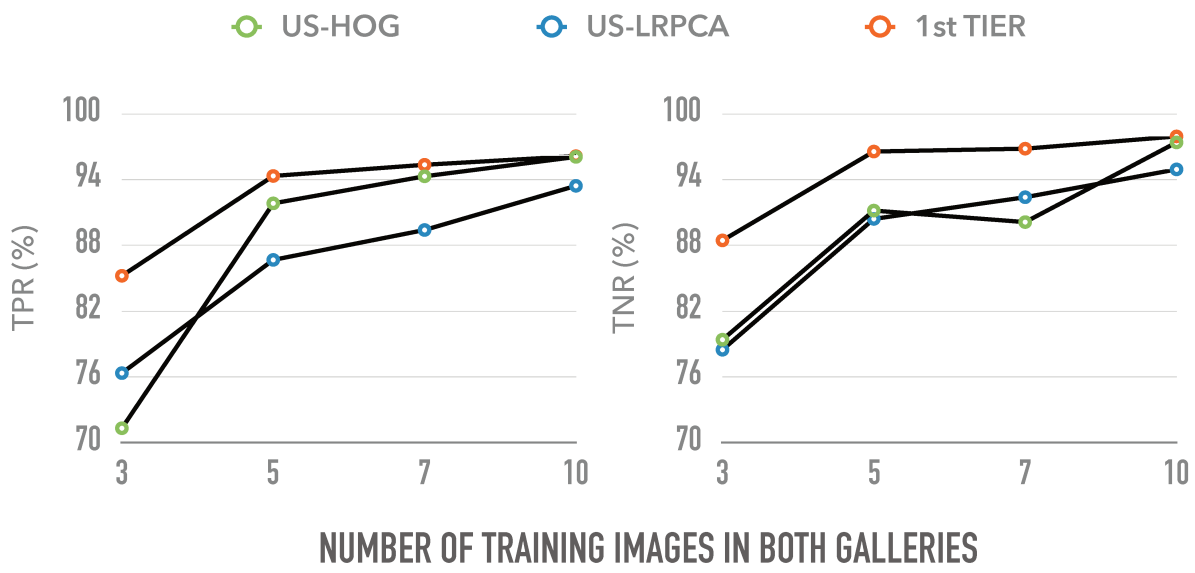
As a preliminary step to filter which images will be analyzed by the 2nd tier, we explored inside-device training of classifiers with hand-crafted features. Instead of tackling the pairwise "same identity or not" task, we use the owner's gallery built during the enrollment to train a user-specific single-image classifier; i.e., to determine if a probe depicts the device owner or not. Aside from the enrollment gallery, referred as the *positive gallery*, a *negative gallery* consisting of face images from different people was embedded into the mobile device to be used in training.

It is important to note that, in order to forward as few images as possible to the 2nd tier, this step must be able to correctly classify the most common authentication scenario; which, in a single-user mobile device, would be the owner trying to access his or her own device. Therefore, we want this step to have a high TPR.

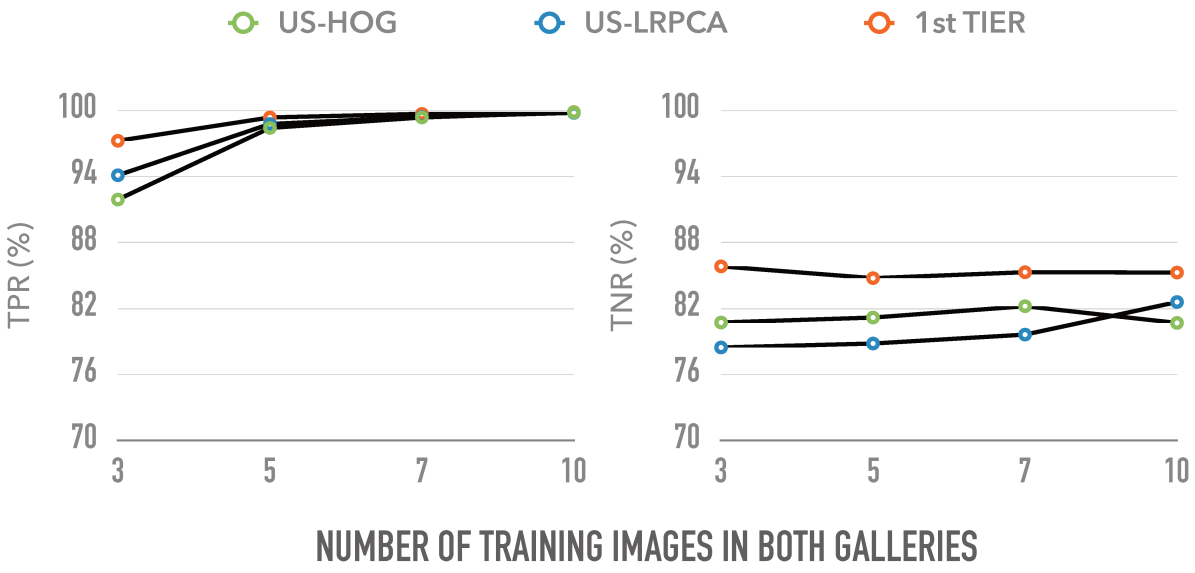
For this experiments, we randomly sampled a negative gallery from the images of MOT, UVAD and OULU-Train. For each pair of RCD-Test and OULU-Test, we considered its gallery as positive images and trained a linear SVM. Once trained, the probe of the pair was classified. Every experiment was performed 20 times — each one with a new negative images — and the annotated accuracy, TPR and TNR is the average among them. These steps were repeated using pairs from RCD-Validation in order to find the best hyperparameters for the SVM.

In Fig. 5.1, we explore how the size of the gallery affects the performance of the classifiers, varying from 3 to 10 images per gallery. We chose to limit to a 10-image gallery, since it is the amount of images being captured during enrollment. Similar to the multiview experiments (Table 5.1), as the gallery size increases, both methods, and consequently their fusion, perform better.

Since the setup with 10 images in each gallery achieved the highest TPR, we fixed the positive gallery size and, in Fig. 5.2, we explored how the user-specific classifiers behaved



(a) RCD-Test.



(b) OULU-Test.

Figure 5.1: User-specific recognition (balanced training) for (a) RCD-Test and (b) OULU-Test. As the number of images increases the overall performance also improves; besides that, the fusion of HOG and LRPCA outperforms both of them separately.

Table 5.5: Results for the complete 2-tiered method. We also report separate performances for each tier and each method within it.

	RCD-Test			OULU-Test		
Method	**Accuracy (%)**	**TPR (%)**	**TNR (%)**	**Accuracy (%)**	**TPR (%)**	**TNR (%)**
US-HOG	96.73	96.07	97.39	90.26	99.83	80.69
US-LRPCA	94.18	93.43	94.93	91.19	99.78	82.59
1st Tier only	97.04	96.14	97.94	91.14	99.82	82.46
HOG	90.05	85.58	94.52	85.83	81.50	90.14
LRPCA	92.19	95.37	89.00	93.92	99.96	87.89
HFCNN - 112×112	93.67	89.22	98.11	94.79	92.54	97.05
HFCNN - 224×224	93.82	90.89	96.76	97.69	96.72	98.65
2nd Tier only 112×112	95.12	91.44	98.80	95.38	93.52	97.24
2nd Tier only 224×224	95.49	92.55	98.43	98.28	97.69	98.87
2-Tiered Method 112×112	96.51	93.39	99.63	98.26	99.65	96.86
2-Tiered Method 224×224	96.73	93.88	99.58	98.48	99.67	97.29

for bigger negative galleries. As the training involves more negative samples, the SVM decision boundary is moved to better separate the negative class, however allowing some positives to be incorrectly classified.

Considering we aim at the highest TPR possible in this step, we select the balanced setup with 10 images in both galleries to compose the 1st tier step.

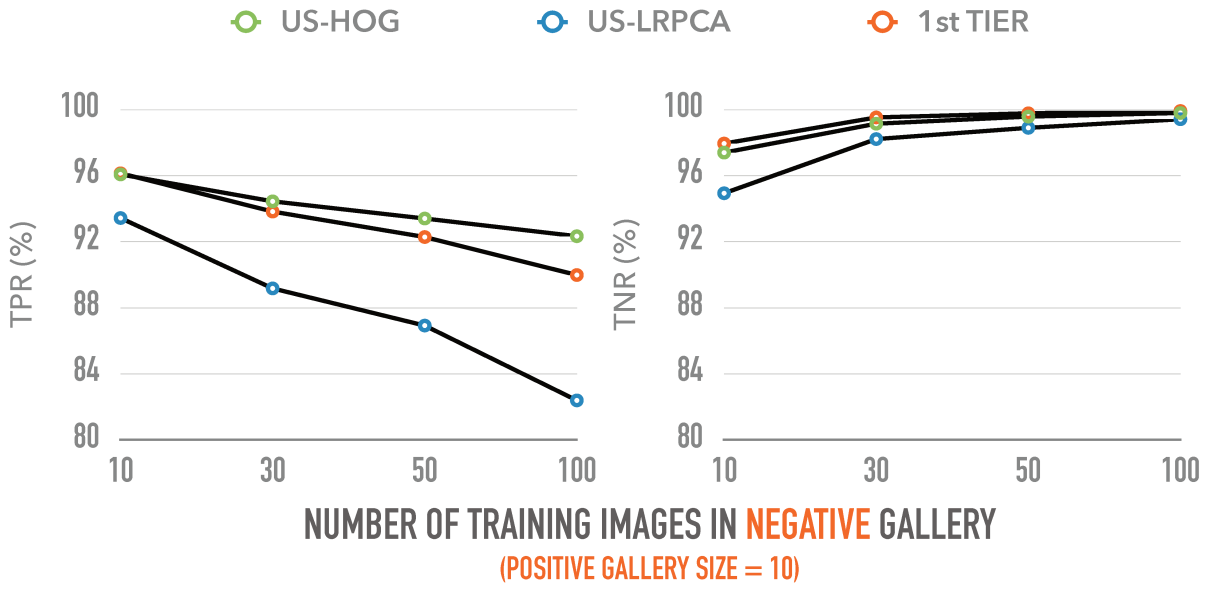
5.4 2-Tiered method

The components explored in previous sections are combined to form the complete 2-tiered method. We use the late fusion of user-specific HOG and LRPCA as the 1st tier's probability and the majority vote of HOG, LRPCA and HFCNN is used as the 2nd tier's decision.

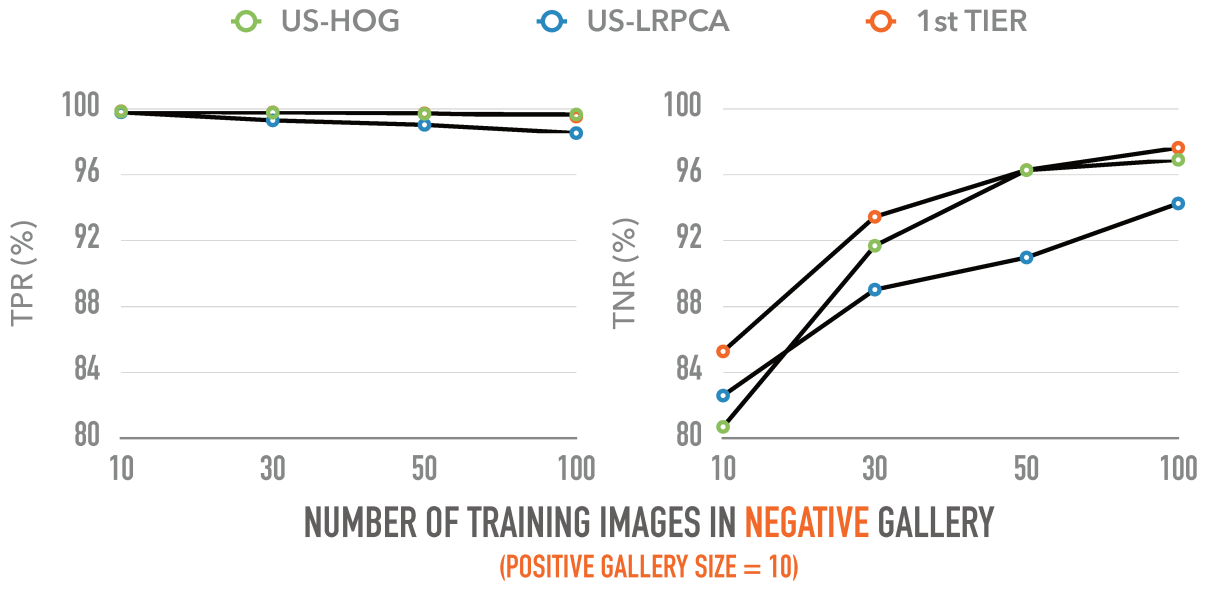
Since the 1st tier (user-specific) acts as a filter step, it is important to define when a probe should be authenticated or denied by that tier's classifiers, or it should proceed to the 2nd tier. This is done by comparing the 1st tier score to two thresholds. If a score falls above the higher threshold, access to the smartphone is allowed, while if it falls below the lower threshold, access is denied; finally, if the score is between both thresholds, then the image is processed by the 2nd tier.

In Table 5.5, we present a summary of the results from the methods that constitute each tier and the complete 2-tiered method. For these experiments, we have selected a higher threshold of 0.7 and a lower threshold of 0.5.

Although the 2-tiered method may not improve accuracy, TPR and TNR so much



(a) RCD-Test.



(b) OULU-Test.

Figure 5.2: User-specific recognition (unbalanced training) for (a) RCD-Test and (b) OULU-Test. With more negative samples in training, the model is able to better classify the negative class, however this reflects in a lower TPR.

when compared to each tier separately, it is important to note the impact of each tier in the complete method. The 1st tier offers an expressive speed-up, since extracting and classifying a probe takes in the order of milliseconds, in comparison to the 0.8 second taken by HFCNN. On the other hand, to achieve a high TPR in this step we have selected the training set proportions between negative and positive images and have strictly limited the amount of images used in training. Naturally, just a few samples from each class are not enough to capture the whole range of facial characteristics from identities distinct from the device owner, nor to be robust to different capture conditions and alterations like illumination, head pose or occlusion.

This strongly reinforces the need of a gallery captured in most distinct capture conditions as possible. While RCD and MOT are diverse datasets, OULU-NPU has very little variation in illumination and head pose, which reflects into homogeneous galleries. Since we selected MOT images to compose the negative spectrum when training the user-specific classifier, most samples from OULU's identities that are different from the classifier's target individual present conditions that approximate them to the positive spectrum. This reflects to a lower TNR for OULU-Test, that is later fixed by the 2nd Tier.

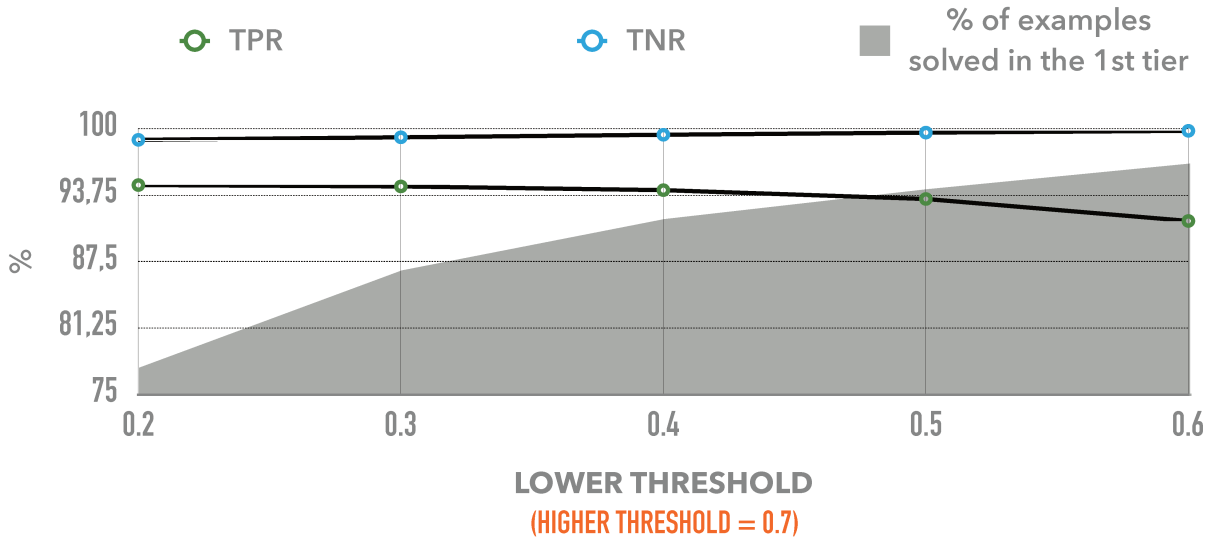
Another important aspect of the 2-tiered method is the higher and lower threshold selection. They provide a simple way to balance the trade-off between speed, TPR and TNR. For example, by increasing the higher threshold it is possible to be stricter when the 1st tier authenticates a probe. This increase false rejections (by lowering TPR), but decrease false acceptances (by increasing TNR).

Besides security, there is also an efficiency aspect related to threshold selection. They also control how many images follow to the 2nd tier and how many are automatically authenticated or denied by the previous step, which directly relates to the overall speed of the solution. Ultimately, we wish most attempts to be dealt with by the 1st tier, while the next step only process those near the 1st tier's decision frontier, where the classifiers have low confidence. Different setup of both thresholds can be offered as an option to the device owner, controlling the trade-off between speed and security. In Fig. 5.3, we expose some threshold setups, the corespondent results for RCD-Test and HFCNN 112×112, and the percentage of samples processed by each tier. A more complete exploration can be found in Appendix A for both RCD-Test and OULU-Test.

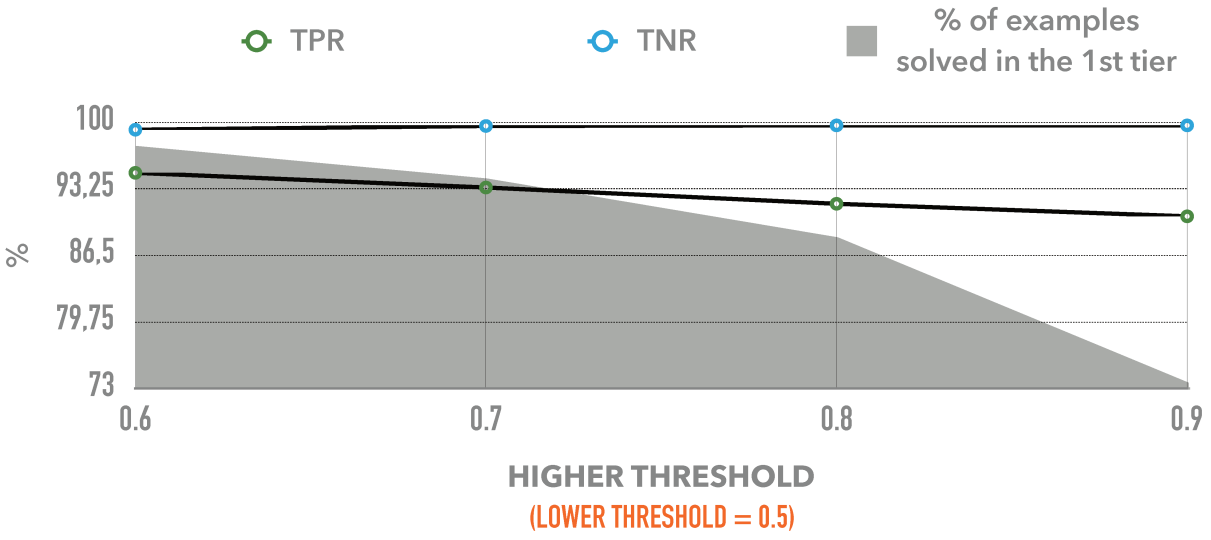
5.5 User-specific threshold learning for the 2nd tier

The user-specific threshold learning was proposed as a way to adapt the methods to images with different characteristics than the ones present during training, while also incorporating information about the device owner into the 2nd tier.

We trained the 2-tiered method with MOT and UVAD datasets, keeping OULU-Train outside training in order to emulate images with unseen characteristics when testing in OULU-Test. With the threshold learning setup proposed in Section 4.4, 200 hybrid images are processed for each testing sample per 2nd tier method, which is very time-consuming for datasets as big as RCD-Test and OULU-Test. In this sense, we tested with 20% of each dataset's samples, randomly selected and balanced for positive and negative classes.



(a) Lower threshold exploration for RCD-Test.



(b) Higher threshold exploration for RCD-Test.

Figure 5.3: Exploration of the (a) lower threshold, while maintaining the higher threshold in 0.7; and (b) higher threshold, while keeping the lower threshold in 0.5. The selection of lower and higher thresholds influences TPR and TNR, as well as the percentage of images that must be analyzed by the 2nd tier (gray area in the graph).

Table 5.6: User-specific threshold learning. The top block of rows presents information regarding the 1st tier, that is not affected by threshold learning; the performance of the 2nd tier with and without threshold learning is shown in the middle and bottom blocks respectively. By learning the 2nd tier acceptance thresholds we manage to adapt individual methods to the image's characteristics from OULU-NPU that were not present during training.

		RCD-Test (20%)			**OULU-Test (20%)**		
	Method	**Accuracy (%)**	**TPR (%)**	**TNR (%)**	**Accuracy (%)**	**TPR (%)**	**TNR (%)**
	US-HOG	96.75	95.90	97.59	90.82	99.88	81.76
	US-LRPCA	94.15	93.37	94.93	90.84	99.78	81.90
	1st Tier	97.01	96.02	98.00	92.36	99.88	84.85
Without Threshold Learning	HOG	90.51	84.47	96.55	77.94	93.64	62.25
	LRPCA	92.42	95.94	88.90	63.74	100.0	27.48
	HFCNN 112×112	93.44	89.18	97.70	89.74	97.03	82.45
	2nd Tier	95.59	93.05	98.14	90.00	98.10	63.89
	2-Tiered Method	96.59	93.70	99.48	93.66	99.76	87.57
With Threshold Learning	HOG	90.25	86.82	93.68	94.99	98.69	91.29
	LRPCA	91.45	88.45	94.45	93.63	99.26	87.99
	HFCNN 112×112	88.27	76.75	99.78	93.15	94.41	91.89
	2nd Tier	93.50	88.18	98.81	96.86	99.32	94.40
	2-Tiered Method	95.55	91.64	99.46	97.67	99.77	95.57

Under these circumstances, we expose in Table 5.6 each tier's components separately and their fusion to achieve the tier performance. Furthermore, both tiers are combined into the final 2-tiered method, that is tested with standard thresholds — a lower threshold of 0.5 and a higher threshold of 0.7. Since the 1st tier (top block of Table 5.6) is not altered by threshold learning, it is shared by both versions of the 2nd tier (middle block of Table 5.6 without learned thresholds — using an acceptance threshold of 0.5 for all three 2nd tier methods — and bottom block with learned thresholds).

When not trained with OULU images, HOG and LRPCA are considerably affected. However, threshold learning is able to adapt the 2nd tier methods, improving the complete solution for OULU-Test dataset. For RCD dataset, the threshold learning has negatively impacted individual methods — in most cases decreasing TPR and slightly increasing TNR due to how the threshold is selected — but this was lessened by the fusion all components and the combination with the 1st tier.

5.6 Comparison with existing methods

Many methods in the literature have approached the facial recognition task either in verification or in identification scenarios. However, only few were focused in the mobile environment, where it is necessary to ponder other factors besides accuracy. For this comparison, we have considered:

- **2-Tiered Method:** both versions of the proposed method, with HFCNN's inputs of size 224×224 and 112×112.

Table 5.7: Comparison of the proposed 2-tiered method with existing methods proposed for face recognition in the literature.

	RCD-Test			OULU-Test		
Method	**Accuracy (%)**	**TPR (%)**	**TNR (%)**	**Accuracy (%)**	**TPR (%)**	**TNR (%)**
2-Tiered Method 112×112	96.51	93.39	99.63	98.26	99.65	96.86
2-Tiered Method 224×224	96.73	93.88	99.58	98.48	99.67	97.29
VGGFace	96.40	94.57	98.23	88.90	83.19	94.61
VGGFace Fine-tuned (Probe, Avg Gal, 0)	92.80	87.66	97.94	94.83	91.74	97.93
ResFace101	92.76	93.84	91.67	91.30	93.77	88.84

Table 5.8: Time and memory analysis for CNNs

		Forward Pass Duration (s)			
Architecture	**Million of Multiply-add**	**Smartphone A**	**Smartphone B**	**Thousand of Parameters**	**Model Size (MB)**
HFCNN 224×224	14,270	4.57	2.43	9,208	35
HFCNN 112×112	3,570	1.15	0.80	9,208	35
VGGFace	15,468	10.27	3.03	134,263	553
ResFace101	7,610	4.31	2.35	64,060	257
MobileNet	574	1.08	0.34	4,230	16
SqueezeNet	388	0.23	0.21	1,230	5
GoogLeNet	1,600	0.97	0.89	6,990	51

- **VGGFace:** we used the network's *fc7* layer as feature extractor. The feature vector of a pair of images consists of the absolute difference and element-wise multiplication of the feature vectors of individual images. A linear SVM trained with it determined the verification outcome. This corresponds to the method presented in the experiments from Table 5.1 without multiview approach.

- **Fine-tuned VGGFace:** the network fine-tuned with hybrid images proposed in Section 4.2.2. This corresponds to the method presented in the experiments from Table 5.2 with hybrid image consisting of the probe and the gallery average image, trained with the same protocol as HFCNN.

- **ResFace101**: a version of the ResNet-101 network, a residual network [33] with 101 layers, fine-tuned for face recognition with CASIA [94] images following the data augmentation described in [58]. For the verification task, we have followed the same steps performed with VGGFace, using the network as a feature extractor and training a SVM with pair's feature vectors.

In Table 5.7, we present the results for the considered methods for RCD-Test and OULU-Test images. The method proposed in this work outperforms, or compares to, the other solutions for both datasets.

We have also chosen state-of-the-art CNNs to compare with the architecture proposed in this work. We present in Table 5.8 the analysis regarding number of operations, pa-

rameters, and also time and memory consumption. Although MobileNet, SqueezeNet and GoogLeNet were designed budgeting number of parameters and operations, they were not trained for the face recognition scenario. Because of that, we did not included them in the comparison presented in Table 5.7.

In comparison to VGGFace, the baseline architecture for this research, we have greatly reduced the number of parameters and performed operations, however HFCNN still need to be further modified to outperform architectures tweaked for efficiency, such as MobileNet and SqueezeNet. A step to improve our network would be to exchange the initial convolutions before the Fire modules to depthwise separable convolutions from Xception architecture [15], further reducing the number of parameters and operations performed, but requiring all layers to be trained from scratch.

5.7 Answering research questions

With the proposed methods and the results exposed in this chapter, we are now able to answer the research questions posed in Section 1.1:

I. **Considering the mobile environment's resource limitations and the computational cost of running a deep network in it, is deep learning a necessary approach for this kind of application?**

   Although the deep networks explored in this work are less efficient than methods based on hand-crafted features, as exposed in Tables 5.8 and 5.4, data-driven models are known to be powerful and robust, capable of overcoming several limitations of feature engineered representations. Despite the gap in efficiency, in our experiments, CNNs have achieved better results than HOG and LRPCA classifiers.

II. **Is it possible to design a deep learning solution for the face verification problem bearing in mind the mobile environment's resource limitations?**

   Yes, the 2-tiered method proposed in this research performs up to par with VGGFace, one of the state-of-the-art methods for face recognition, while greatly reducing processing time and memory footprint. When compared to other CNNs tweaked for efficiency, such as MobileNet or SqueezeNet (networks that were not trained for facial verification), HFCNN is still up to four times slower and its model twice bigger. However it is definitely a viable architecture for mobile real-time applications.

III. **Would a fusion of deep and hand-crafted approaches lead to better accuracy than the methods separately?**

   Yes, our experiments showed that hand-crafted and data-driven techniques can complement each other and their fusion performs better than those methods separately for the task at hand.

IV. **How many images should a user's gallery have and in what resolution should they be?**

Our experiments showed a performance improvement as more images were used, however it may also imply in increasing processing time (e.g., if we need to process each multiview pair every time). We used images in 224×224 and 112×112 resolutions and both HFCNN setups had similar performances, although 112×112 version is 3 times faster.

V. **Is it possible to adapt multi-class networks for the binary verification scenario, without a significant increase in memory and processing time?**

Yes, by using hybrid images we were able to encode information of the probe and gallery images into a single one without increasing time and memory usage. With this approach, no architectural modification is necessary to adapt a network for the verification task.

VI. **Would user-specific information improve our solution? If so, how should it be incorporated?**

Not only it improved accuracy, but by adding a user-specific classification with fast-extraction features we were able to manage the method's bottleneck regarding processing time. In addition to that, learning a user-specific threshold for a user-independent task, such as "same identity or not" verification, also improved performance.

Chapter 6

Conclusion and Future Work

In this research, we have proposed a method for facial verification optimized for the mobile environment. A real-time application in this scenario needs to take into account factors such as memory usage, unstable connectivity, battery consumption and limited processing power in order to impact neither its own performance nor the system as a whole.

The designed method consists of a 2-tiered procedure that combine hand-crafted and data-driven features to verify if the person present in a picture corresponds to the device owner. The 1st tier employs the fusion of two user-specific linear SVMs trained on HOG and LRPCA features extracted directly from the user's gallery images. This training is done inside the device once the enrollment process is concluded and is focused on authenticating with a high TPR. Whereas the 2nd tier fuses the results of a CNN and two logistic regression classifiers trained on HOG and LRPCA features. Contrary to the previous tier, this step's techniques aim to check if two images depict the same identity, without considering information about the device owner. Lastly, we adjust the acceptance threshold of the 2nd tier's classifiers with pairs of images constructed with the owner's gallery in order to better adjust this last step to his or her characteristics.

One of the main contributions of this work, the Hybrid-Fire CNN architecture was inspired on VGGFace [64] and SqueezeNet [40] and was able to perform a par with VGGFace, but with a model 16 times smaller and 4 times faster. Besides architectural adjusts thought directly for the mobile environment, HFCNN uses hybrid images that combine the information of multiple face pictures into one, as a way to limit the necessity of multiple forward passes. Hybrid images can also be viewed as a simple way to adapt a multi-class formulation, in our case face identification in a domain with multiple identities, to a binary verification formulation, identifying if two pictures belong to the same identity.

In addition to these contributions, we have also collected a new dataset focused on selfie pictures[1]. The RECOD Selfie Dataset comprises 2873 images from 56 individuals with varied capture conditions regarding illumination, head pose, partial occlusion, background and facial expression.

Many research paths related to what have been done in this work can be explored in future works. The proposed 2-tiered solution can be extended to the identification task,

[1] dx.doi.org/10.6084/m9.figshare.5427142

where we need to relate a probe to a single identity among all present in a database. In this scenario, it is not viable to compare a probe to the gallery of all possible identities for a real-time application. In this case, an alternative to decrease processing time would be to compare the probe with its cohorts, i.e., groups of identities that have similar characteristics.

Additionally, we would like to further lighten HFCNN architecture by exploring a more extensive use of depthwise separable convolutions [15], as a mean to decrease the number of operations performed and consequently speed up the network. Although our model is considerably small, deep compression and network pruning methods could still be used to reduce it even more.

Regarding the hybrid image formulation, we believe that it is possible to improve recognition by adding relevant information to its third channel. Besides that, this formulation can also be applied to other tasks and problems, e.g., composing the hybrid image with a channel for each of two consecutive frames and information related to their changes for a video task.

Finally, we would like to explore different ways to compose the gallery, in order to select — during enrollment or update of the gallery — the most discriminative images among the full set and discard those that are not useful to the verification.

Bibliography

[1] T. Ahonen, A. Hadid, and M. Pietikainen. Face description with local binary patterns: Application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):2037–2041, 2006.

[2] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.

[3] Y. Bengio. Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[4] Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.

[5] Luca Bondi, Luca Baroffio, Matteo Cesana, Marco Tagliasacchi, G Chiachia, and Anderson Rocha. Rate-energy-accuracy optimization of convolutional architectures for face recognition. *Journal of Visual Communication and Image Representation*, 36:142–148, 2016.

[6] Danyl Bosomworth. Mobile marketing statistics 2015. *Leeds: Smart Insights (Marketing Intelligence) Ltd*, 2015.

[7] 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*, To appear.

[8] Z. Boulkenafet, J. Komulainen, Lei. 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*, 2017.

[9] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a 'siamese' time delay neural network. In *Advances in neural information processing systems*, pages 737–744, 1994.

[10] AT&T Laboratories Cambridge. *The Database of Faces*. `http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html`, May, 2017.

[11] Pierluigi Carcagnì, Marco Del Coco, Pier Luigi Mazzeo, Andrea Testa, and Cosimo Distante. Features descriptors for demographic estimation: a comparative study. In *International Workshop on Video Analytics for Audience Measurement in Retail and Digital Signage*, pages 66–85, 2014.

[12] Rama Chellappa, Pawan Sinha, and P Jonathon Phillips. Face recognition by computers and humans. *IEEE Computer*, 43(2), 2010.

[13] Giovani Chiachia. *Learning person-specific face representations*. PhD thesis, State University of Campinas, 2013.

[14] Kwontaeg Choi, Kar-Ann Toh, and Hyeran Byun. Realtime training on mobile devices for face recognition applications. *Pattern Recognition*, 44(2):386–400, 2011.

[15] François Chollet. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint arXiv:1610.02357*, 2016.

[16] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 539–546, 2005.

[17] D. Cox and N. Pinto. Beyond simple features: A large-scale feature search approach to unconstrained face recognition. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 8–15, 2011.

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

[19] Zhen Cui, Wen Li, Dong Xu, Shiguang Shan, and Xilin Chen. Fusing robust face region descriptors via multiple metric learning for face recognition in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3554–3561, 2013.

[20] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 886–893, 2005.

[21] Kresimir Delac and Mislav Grgic. A survey of biometric recognition methods. In *IEEE International Symposium on Electronics in Marine*, pages 184–193, 2004.

[22] Misha Denil, Babak Shakibi, Laurent Dinh, Nando de Freitas, et al. Predicting parameters in deep learning. In *Advances in neural information processing systems*, pages 2148–2156, 2013.

[23] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in neural information processing systems*, pages 1269–1277, 2014.

[24] Hossein Falaki, Ratul Mahajan, Srikanth Kandula, Dimitrios Lymberopoulos, Ramesh Govindan, and Deborah Estrin. Diversity in smartphone usage. In *ACM International Conference on Mobile Systems, Applications and Services*, pages 179–194, 2010.

[25] TechCrunch Fitz Tepper. *MasterCard launches its 'selfie pay' biometric authentication app in Europe*. `https://techcrunch.com/2017/09/12/face-id-is-replacing-touch-id-on-the-new-iphone-x/`, September, 2017.

[26] William T Freeman and Michal Roth. Orientation histograms for hand gesture recognition. In *International Workshop on Automatic Face and Gesture Recognition*, volume 12, pages 296–301, 1995.

[27] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

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

[29] David Gschwend. *Netscope CNN Analyzer*. `http://dgschwend.github.io/netscope/quickstart.html`, June, 2017.

[30] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[31] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[32] Kaiming He and Jian Sun. Convolutional neural networks at constrained time cost. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5353–5360, 2015.

[33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[34] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *MIT Neural Computation*, 18(7):1527–1554, 2006.

[35] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[36] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[37] Fu Jie Huang, Y-Lan Boureau, Yann LeCun, et al. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[38] Gary B Huang, Honglak Lee, and Erik Learned-Miller. Learning hierarchical representations for face verification with convolutional deep belief networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2518–2525, 2012.

[39] Gary B Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, University of Massachusetts, Amherst, 2007.

[40] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. *arXiv preprint arXiv:1602.07360*, 2016.

[41] A. Jain, L. Hong, and S. Pankanti. Biometric identification. *Communications of the ACM*, 43(2):90–98, 2000.

[42] A. K. Jain, R. Bolle, and S. Pankanti. *Biometrics: personal identification in networked society*. Springer US, 2006.

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

[44] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[45] T. Kanade. *Picture processing system by computer complex and recognition of human faces*. PhD thesis, Kyoto University, 1973.

[46] Ira Kemelmacher-Shlizerman, Steven M Seitz, Daniel Miller, and Evan Brossard. The megaface benchmark: 1 million faces for recognition at scale. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4873–4882, 2016.

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

[48] 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.

[49] Yann LeCun, John S Denker, Sara A Solla, Richard E Howard, and Lawrence D Jackel. Optimal brain damage. In *Advances in neural information processing systems*, volume 2, pages 598–605, 1989.

[50] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[51] S. Z. Li and A. K. Jain. *Handbook of face recognition*. Springer-Verlag London, 2 edition, 2011.

[52] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[53] TIME Lisa Eadicicco. *Americans Check Their Phones 8 Billion Times a Day*. `http://time.com/4147614/smartphone-usage-us-2015/`, June, 2017.

[54] Chengjun Liu and Harry Wechsler. A shape-and texture-based enhanced fisher classifier for face recognition. *IEEE Transactions on Image Processing*, 10(4):598–608, 2001.

[55] David G Lowe. Distinctive image features from scale-invariant keypoints. *Internatinal Journal of Computer Vision*, 60(2):91–110, 2004.

[56] J. Lu, V. E. Liong, X. Zhou, and J. Zhou. Learning compact binary face descriptor for face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(10):2041–2056, 2015.

[57] Aleix M Martinez. The AR face database. *CVC Technical Report*, 24, 1998.

[58] Iacopo Masi, Anh Tran, Tal Hassner, Jatuporn Toy Leksut, and Gérard Medioni. Do We Really Need to Collect Millions of Faces for Effective Face Recognition? In *European Conference on Computer Vision*, 2016.

[59] B Miller. Everything you need to know about biometric identification. personal identification news 1988 biometric industry directory. washington dc: Warfel & miller. *Inc., Washington DC*, 1988.

[60] Cisco Mobile. *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021*. `http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html`, June, 2017.

[61] TechCrunch Natasha Lomas. *MasterCard launches its 'selfie pay' biometric authentication app in Europe*. `https://techcrunch.com/2016/10/04/mastercard-launches-its-selfie-pay-biometric-authentication-app-in-europe/`, September, 2017.

[62] Engadget Nick Summers. *Mastercard's 'selfie pay' comes to Europe*. `https://www.engadget.com/2016/10/04/mastercard-online-selfie-pay-europe/`, September, 2017.

[63] Margarita Osadchy, Yann Le Cun, and Matthew L Miller. Synergistic face detection and pose estimation with energy-based models. *Journal of Machine Learning Research*, 8(May):1197–1215, 2007.

[64] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In *British Machine Vision Conference*, volume 1, page 6, 2015.

[65] Saffe Payments. *Saffe, your money, your face*. `http://www.saffe.com.br/`, September, 2017.

[66] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[67] Alex Pentland and Tanzeem Choudhury. Face recognition for smart environments. *IEEE Computer*, 33(2):50–55, 2000.

[68] P. J. Phillips, J. R. Beveridge, B. A. Draper, G. Givens, A. J. O'Toole, D. S. Bolme, J. Dunlop, Y. M. Lui, H. Sahibzada, and S. Weimer. An introduction to the good, the bad, & the ugly face recognition challenge problem. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 346–353, 2011.

[69] P Jonathon Phillips, Hyeonjoon Moon, Syed A Rizvi, and Patrick J Rauss. The FERET evaluation methodology for face-recognition algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10):1090–1104, 2000.

[70] A. Pinto, W. R. Schwartz, H. Pedrini, and A. Rocha. Using visual rhythms for detecting video-based facial spoof attacks. *IEEE Transactions on Information Forensics and Security*, 10(5):1025–1038, 2015.

[71] GitHub repository. *Porting Caffe to Android platform*. `https://github.com/sh1r0/caffe-android-lib`, June, 2017.

[72] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[73] 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. *Internatinal Journal of Computer Vision*, 115(3):211–252, 2015.

[74] 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.

[75] T. Serre, G. Kreiman, M. Kouh, C. Cadieu, U. Knoblich, and T. Poggio. A quantitative theory of immediate visual recognition. *Progress in Brain Research*, 165:33–56, 2007.

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

[77] Nitish Srivastava, Geoffrey 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.

[78] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep convolutional network cascade for facial point detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3476–3483, 2013.

[79] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv:1602.07261*, 2016.

[80] 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.

[81] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[82] 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.

[83] Xiaoyang Tan and Bill Triggs. Enhanced local texture feature sets for face recognition under difficult lighting conditions. *IEEE Transactions on Image Processing*, 19(6):1635–1650, 2010.

[84] Gerald Tesauro. Practical issues in temporal difference learning. *Machine learning*, 8(3-4):257–277, 1992.

[85] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[86] Berkeley Vision and Learning Center. *AlexNet Caffe Model*. `https://github.com/BVLC/caffe/tree/master/models/bvlc_alexnet`, May, 2017.

[87] Berkeley Vision and Learning Center. *Caffe Model Zoo*. `https://github.com/BVLC/caffe/wiki/Model-Zoo`, May, 2017.

[88] Berkeley Vision and Learning Center. *GoogLeNet Caffe Model*. `https://github.com/BVLC/caffe/tree/master/models/bvlc_googlenet`, May, 2017.

[89] University of Oxford Visual Geometry Group. *VGGFace Caffe Model*. `http://www.robots.ox.ac.uk/~vgg/software/vgg_face/`, May, 2017.

[90] J Wayman. A definition of biometrics. *National Biometric Test Center Collected Works*, 1:21–23, 2000.

[91] J. Wayman, A. K. Jain, D. Maltoni, and D. Maio. An introduction to biometric authentication systems. *Biometric Systems*, pages 1–20, 2005.

[92] L. Wiskott, J.-M. Fellous, N. Kuiger, and C. Von Der Malsburg. Face recognition by elastic bunch graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):775–779, 1997.

[93] Lior Wolf, Tal Hassner, and Itay Maoz. Face recognition in unconstrained videos with matched background similarity. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 529–534, 2011.

[94] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*, 2014.

[95] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[96] Wenyi Zhao, Rama Chellappa, P Jonathon Phillips, and Azriel Rosenfeld. Face recognition: A literature survey. *ACM Computing Surveys*, 35(4):399–458, 2003.

Appendix A

Higher and Lower Thresholds Exploration

As seen in Section 5.4, two thresholds determine when a probe analyzed by the 1st tier should be authenticated, denied or processed by the following tier. Since the 2nd tier is computationally more expensive, we want most probes to be processed and filtered by the initial tier; therefore the threshold selection also controls the equilibrium between TPR, TNR and the overall speed of the complete method.

Although a strategy similar to the user-specific threshold learning (Section 4.4) could be devised to automatically determine both thresholds, we believe that an optimal trade-off between efficiency and security is personal to each user and depends on how the device is used. A user might prefer a faster authentication; other might value security more, while a third may choose an in-between option. Because of that, we suggest that multiple scenarios (threshold setups) should be offered to the users, allowing them to choose the one that better fits their needs.

In this chapter, we explore different setups and how they affect TPR, TNR and the percentage of images that are analyzed by 2nd tier. We present this exploration for RCD-Test in Table A.1 and for OULU-Test in Table A.2, considering the 2-tiered method with 112×112 input size for HFCNN. We remark that part of Table A.1 was already presented in Fig. 5.3.

It is important to point out some characteristics that are related to the process of selecting the thresholds:

- **The method speed is controlled by the difference between higher and lower thresholds.** A small gap between them means most authentication attempts are only analyzed by the 1st tier, whereas, when this gap increases, more probes are forwarded to the method's second tier.

- **The higher threshold determines how rigorous or tolerant the 1st tier is to authenticate a probe.** A smaller value requires a less confident prediction that a probe depicts the device owner, increasing TPR and lowering TNR. A threshold near 1.0 enforces a higher confidence, causing the opposite behavior on TPR and TNR.

- **The lower threshold, on the other hand, establishes the 1st tier's behavior to deny a probe.** A lower value (near 0.0) requires a strong confidence that a probe does not belong to the device owner to be automatically denied in this tier. This means more images will be forwarded to the 2nd tier. Whereas a value near 0.5 tends to deny more images, increasing TNR but decreasing TPR.

Table A.1: Higher and lower thresholds exploration for RCD-Test with HFCNN 112×112.

Higher Threshold	**Lower Threshold**	**ACC (%)**	**TPR (%)**	**TNR (%)**	**% of examples in 1st tier**	**% of examples in 2nd tier**
0.6	0.2	97.38	96.19	98.57	80.80	19.20
	0.3	97.43	96.06	98.80	89.95	10.05
	0.4	97.38	95.71	99.05	94.78	5.22
	0.5	97.06	94.88	99.25	97.60	2.40
	0.6	96.11	92.81	99.42	100.00	0.00
0.7	0.2	96.82	94.70	98.95	77.51	22.49
	0.3	96.88	94.57	99.19	86.66	13.34
	0.4	96.83	94.22	99.43	91.49	8.51
	0.5	96.51	93.39	99.63	94.30	5.70
	0.6	95.56	91.32	99.80	96.71	3.29
0.8	0.2	96.03	93.05	99.01	71.54	28.46
	0.3	96.09	92.92	99.25	80.69	19.31
	0.4	96.04	92.58	99.49	85.52	14.48
	0.5	95.72	91.74	99.69	88.33	11.67
	0.6	94.77	89.67	99.86	90.74	9.26
0.9	0.2	95.40	91.78	99.01	56.80	43.20
	0.3	95.45	91.65	99.25	65.95	34.05
	0.4	95.40	91.31	99.50	70.78	29.22
	0.5	95.08	90.47	99.69	73.59	26.41
	0.6	94.13	88.40	99.86	76.00	24.00

Table A.2: Higher and lower thresholds exploration for OULU-Test with HFCNN 112×112.

Higher Threshold	**Lower Threshold**	**ACC (%)**	**TPR (%)**	**TNR (%)**	**% of examples in 1st tier**	**% of examples in 2nd tier**
0.6	0.2	95.53	99.78	91.27	68.18	31.82
	0.3	95.58	99.78	91.39	81.21	18.79
	0.4	95.80	99.78	91.82	89.95	10.05
	0.5	96.09	99.78	92.41	96.00	4.00
	0.6	96.47	99.76	93.18	100.00	0.00
0.7	0.2	97.69	99.66	95.72	65.45	34.55
	0.3	97.75	99.66	95.84	78.48	21.52
	0.4	97.96	99.66	96.27	87.21	12.79
	0.5	98.26	99.65	96.86	93.26	6.74
	0.6	98.63	99.63	97.63	97.26	2.74
0.8	0.2	98.30	99.49	97.11	63.80	36.20
	0.3	98.36	99.49	97.23	76.83	23.17
	0.4	98.57	99.49	97.66	85.57	14.43
	0.5	98.87	99.49	98.25	91.62	8.38
	0.6	99.24	99.47	99.02	95.62	4.38
0.9	0.2	97.26	97.27	97.24	54.85	45.15
	0.3	97.31	97.27	97.35	67.88	32.12
	0.4	97.53	97.27	97.78	76.62	23.38
	0.5	97.82	97.27	98.38	82.67	17.33
	0.6	98.20	97.25	99.15	86.67	13.33