Chapter 12
Deep Triplet Embedding Representations for Liveness Detection

Federico Pala and Bir Bhanu

Abstract Liveness detection is a fundamental element for all biometric systems that have to be safe against spoofing attacks at sensor level. In particular, for an attacker it is relatively easy to build a fake replica of a legitimate finger and apply it directly to the sensor, thereby fooling the system by declaring its corresponding identity. In order to ensure that the declared identity is genuine and it corresponds to the individual present at the time of capture, the task is usually formulated as a binary classification problem, where a classifier is trained to detect whether the fingerprint at hand is real or an artificial replica. In this chapter, unlike the binary classification model, a metric learning approach based on triplet convolutional networks is proposed. A representation of the fingerprint images is generated, where the distance between feature points reflects how much the real fingerprints are dissimilar from the ones generated artificially. A variant of the triplet objective function is employed, that considers patches taken from two real fingerprint and a replica (or two replicas and a real fingerprint), and gives a high penalty if the distance between the matching couple is greater than the mismatched one. Given a test fingerprint image, its liveness is established by matching its patches against a set of reference genuine and fake patches taken from the training set. The use of small networks along with a patch-based representation allows the system to perform the acquisitions in real time and provide state-of-the-art performances. Experiments are presented on several benchmark datasets for liveness detection including different sensors and fake fingerprint materials. The approach is validated on the cross-sensor and cross-material scenarios, to understand how well it generalizes to new acquisition devices, and how robust it is to new presentation attacks.

F. Pala (✉) · B. Bhanu
Center for Research in Intelligent Systems - University of California,
Riverside Winston Chung Hall Suite 216, 900 University Avenue,
Riverside, CA 92521-0425, USA
e-mail: fedpala@ucr.edu

B. Bhanu
e-mail: bhanu@cris.ucr.edu

B. Bhanu and A. Kumar (eds.), *Deep Learning for Biometrics*,
Advances in Computer Vision and Pattern Recognition,
DOI 10.1007/978-3-319-61657-5_12

12.1 Introduction

Biometrics authentication systems have become part of the daily routine for millions of people around the world. A large number of people use their fingerprints every day in order to pass the security checkpoints at airports, to access their personal mobile devices [1] and to access restricted areas. The popularity of this biometric with respect of others such as face and iris recognition lies on its reliability, that has been proven during the last decades, its implementation at an affordable cost, and especially on the simplicity of touching a surface to get immediately authenticated.

Unfortunately, all these advantages come with various security and privacy issues. Different attacks can be performed to the authentication system in order to grant access to some exclusive area or to steal confidential data. For instance, the software and the network configuration can have security holes or bugs, and the matching algorithms can be fooled if the attacker knows the software implementation details [2]. Moreover, whereas a physical key or badge can be replaced, fingerprints are permanent and the pattern on their surface can be easily captured and reproduced. It can be taken from a high-resolution photograph or from a print left on a surface such as a mug or even a piece of paper. A high-quality reproduction of the pattern on some gummy material can be simply applied to the scanner [3] so that the attacker can fool the authentication system by declaring its corresponding identity. Since the sensor device is inevitably at a direct contact of the user being captured, it is considered one of the weakest point in the entire biometrics system. Because of this, there is a growing interest in automatically analyzing the acquired fingerprint images in order to catch potential malignant users [4]. This kind of attacks are known as presentation attacks [5], and liveness detection techniques are designed to spot them by formulating a binary classification problem with the aim of establishing whether a given biometrics comes from the subject present at the time of capture [6].

The liveness of a fingerprint can be established by designing a software system that analyzes the same images used by the recognition algorithm, or by equipping the scanner with additional hardware. These last prevention measures are called hardware-based systems [7] and are generally more accurate since they take advantage of additional cues. Anyway, the software of a fingerprint scanner can be updated with no additional cost, and if a software technique is robust to a variety of attacks and does not annoy the users with too many false positives, it can be an alternative with regard to acquiring new sensing devices.

Recently, different studies [8–10] have shown the effectiveness of deep learning algorithms for the task of fingerprint liveness detection. Deep learning has seriously improved the state of the art in many fields such as speech recognition, natural language processing, and object recognition [11–13]. The ability to generate hierarchical representations and discover complex structures in raw images allows for better representations with respect to traditional methods based on handcrafted features. Software-based systems for fingerprint liveness detection can take advantage of the very broad literature where similar tasks have already been addressed. Among the recent works, we noticed that it has not yet directly modeled a notion of similarity among real and fake fingerprints that can capture the underlying factors that explain their inter- and intra-class variations. We make a step in this direction by proposing a deep metric learning approach based on Triplet networks [14]. Specifically, these networks map the fingerprint images into a representation space, where the learned distance captures the similarity of the examples coming from the same class and push away the real samples from the fake ones. Unlike other metric learning approaches, such as the ones involving Siamese networks [15], the triplet objective function puts in direct comparison the relation among the classes, giving a notion of context that does not require a threshold selection in order make the prediction [14].

We propose a framework that learns a representation from fingerprint patches, starting from a set of real and fake samples given as a training set. Since at test time only a fingerprint image is given, we make our decision on the basis of a matching score, computed against a set of real and fake patches given as a reference. The similarity metric is learned using an improved version [16, 17] of the original triplet objective formulation [14] that adds a pairwise term that more firmly forces the closeness of two examples of the same class. We performed extensive experiments using ten datasets taken from the fingerprint liveness detection competitions LivDet[1] organized by the Department of Electrical and Electronic Engineering of the University of Cagliari, in cooperation with the Department of Electrical and Computer Engineering of the Clarkson University, held in the years 2009 [7], 2011 [18] and 2013 [19]. We compare our approach with respect to the state of the art, getting competitive performance for all the examined datasets. We also perform the cross-dataset and cross-material experiments, in order to evaluate if the obtained fingerprint representation can be reused in different settings or to spot materials that have not been seen during training.

The chapter is structured as follows. In Sect. 12.2 we present some of the current approaches for designing fingerprint liveness detection systems, and the current state of the art. In Sect. 12.3 we explain the details of the proposed framework and in Sect. 12.4 we provide the experimental results. The final Sect. 12.5 is dedicated to the conclusions.

12.2 Background and Previous Work

In this section, we describe various fingerprint liveness detection techniques, particularly with a focus on the ones related to our method, which can be considered as a static software-based technique [6]. Subsequently, we provide details on some of the most recently proposed software-based approaches in order to contextualize our method and highlight our contributions.

[1]http://livdet.org/.

12.2.1 Background

A first categorization of liveness detection systems can be made by distinguishing between hardware and software systems. As already mentioned, hardware-based systems, also called sensor-based [6], use additional information in order to spot the characteristics of living human fingerprints. Useful clues can be found for instance by detecting the pattern of veins underlying the fingerprints surface [20], measuring the pulse [21], by performing odor analysis [22] and by employing near-infrared sensors [23].

Software, also called feature-based systems, are algorithms that can be introduced into a sensor software in order to add the liveness detection functionality. Our approach falls in this category and can also be considered static, to distinguish it from methods that use multiple images taken during the acquisition process. For instance, dynamic methods can exploit the distortion of the fingertip skin since with respect to gummy materials, it differs in terms of flexibility. In the approach proposed by [24], the user is asked to rotate the finger while touching the sensor. The distortion of the different regions at a direct contact of the sensor are characterized in terms of optical flow and the liveness prediction is based on matching the encoded distortion code sequences over time.

In order to design a software-based system based on machine learning algorithms, a database of real and fake examples of fingerprints is needed. The more spoofing techniques are used, the more the algorithm will be able to generalize to new kind of attacks. In a second categorization of liveness detection systems, we consider how the fingertip pattern is taken from the victim. In the cooperative method, the victim voluntarily puts his/her finger on some workable material that is used to create a mold. From this mold, it is possible to generate high-quality reproductions by filling it with materials such as gelatin, silicone, and wooden glue. Figure 12.1 shows some photographs of artificial fingers. Noncooperative methods instead, capture the scenarios where the pattern has been taken from a latent fingerprint. After taking a high-resolution picture, it is reproduced by generating a three-dimensional surface,

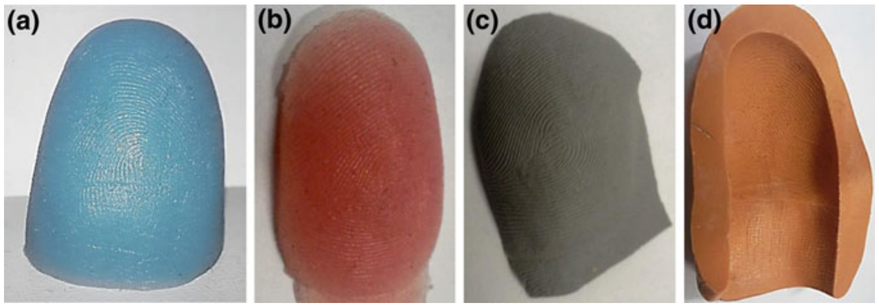

Fig. 12.1 Examples of artificial finger replicas made using different silicon rubber materials: **a** GLS, **b** Ecoflex, **c** Liquid Ecoflex and **d** a Modasil mold

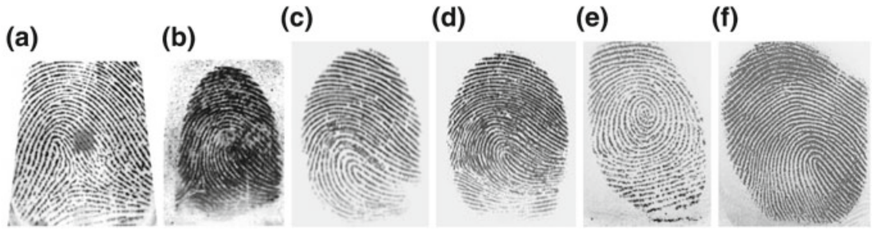

Fig. 12.2 Examples of fake acquisitions from the LivDet 2011 competition (cooperative). From Biometrika **a** Latex, **b** Silicone, from Digital **c** Latex, **d** Gelatine, from Sagem **e** Silicone, **f** Play-doh

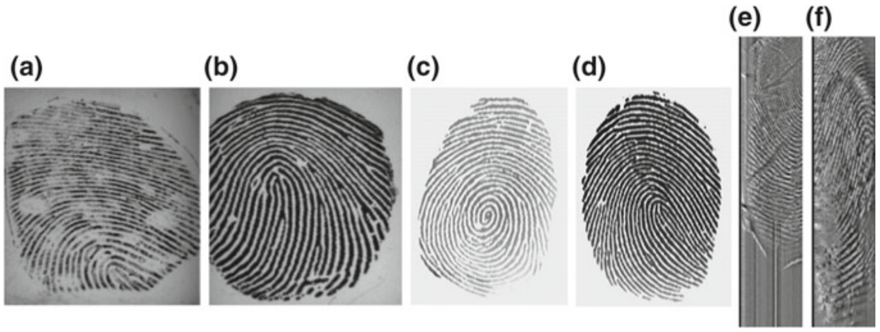

Fig. 12.3 Examples of fake acquisitions from the LivDet 2013 competition. Noncooperative: from Biometrika **a** Gelatine, **b** Wooden Glue, from Italdata **c** Ecoflex, **d** Modasil. Cooperative: from Swipe **e** Latex, **f** Wooden Glue

for instance, by printing it into a circuit board. At this point, a mold can be generated and filled with the above-mentioned materials. The quality of images is inferior as compared to the cooperative methods, and usually, software-based systems have better performance on rejecting these reproductions. Figures 12.2 and 12.3 show several acquisitions, where the fingertip pattern has been captured using the cooperative and noncooperative methods.

12.2.2 Previous Work

In this subsection, we discuss some of the previous work on static software-based fingerprint liveness detection systems. We start by presenting some hand crafted feature-based approaches and conclude with the more recently proposed deep learning techniques.

One of the first approaches to fingerprint liveness detection has been proposed by [25]. It is based on the perspiration pattern of the skin that manifests itself into static and dynamic patterns on the dielectric mosaic structure of the skin. The classification is based on a set of measures extracted from the data and classified using a neural network. In [26], the same phenomenon is captured by a wavelet transform applied to the ridge signal extracted along the ridge mask. After extracting a set of measures from multiple scales, the decision rule is based on classification trees.

Other approaches build some descriptors from different kind of features that are conceived specifically for fingerprint images. In [27] Local binary pattern (LBP) histograms are proposed along with wavelet energy features. The classification is performed by a hybrid classifier composed of a neural network, a support vector machine (SVM) and a k-nearest neighbor classifier. Similar to LBP, the Binary Statistical Image Features (BSIF) [28], encode local textural characteristics into a feature vector and SVM is used for classification. In [29] a Local Contrast Phase Descriptor (LCPD) is extracted by performing image analysis in both the spatial and frequency domains. The same authors propose the use of the Shift-Invariant Descriptor (SID) [30], originally introduced by [31], which provides rotation and scale invariance properties. SID, along with LCPD provides competitive and robust performances on several datasets.

Recently, deep learning algorithms have been applied to fingerprint liveness detection with the aim of automatically finding a hierarchical representation of fingerprints directly from the training data. In [9] the use of convolutional networks has been proposed. In particular, the best results [9] are obtained by fine-tuning the AlexNet and VGG architectures proposed by [11, 32], previously trained on the Imagenet dataset of natural images [33]. From their experimental results, it seems that the factors that most influence the classification accuracy are the depth of the network, the pretraining and the data augmentation they performed in terms of random crops. Since we use a patch-based representation we employ a smaller, but reasonably deep architecture. The use of patches does not require resizing all the images to a fixed dimension, and at the same time, the number of examples is increased so that pretraining can be avoided.

In [8], deep representations are learned from fingerprint, face, and iris images. They are used as traditional features and fed into SVM classifiers to get a liveness score. The authors focus on the choice of the convolutional neural network parameters and architecture.

In [34] deep Siamese networks have been considered along with classical pretrained convolutional networks. This can be considered the most similar work to this chapter since they also learn a similarity metric between a pair of fingerprints. However, their use of metric learning is different since they assume a scenario where the enrollment fingerprint is available for each test image. That is, the decision is made by comparing fingerprints of the same individual. Our approach instead is more versatile and can be applied even if the enrollment fingerprint image is not available.

Different from the above studies, [10, 35] do not give the entire image to the deep learning algorithm but extract patches from the fingerprint acquisition after removing the background. [35] uses classical ConvNets with a binary cross-entropy loss, along with a majority voting scheme to make the final prediction. [10] proposes deep belief networks and use contrastive divergence [36] for pretraining and fine-tunes on the real and fake fingerprint images. The decision is based on a simple threshold applied to the output of the network. Our work is substantially different because it proposes a framework where triplet architectures are used along with a triplet and pairwise objective function.

Summarizing, the contribution of this chapter are (i) a novel deep metric learning based framework, targeted to fingerprint liveness detection, able to work in real time with state-of-the-art performance, (ii) a patch-based and fine-grained representation of the fingerprint images that makes it possible to train a reasonably deep architecture from scratch, even with few hundreds of examples, and that shows superior performance even in settings different from the ones used for training.

12.3 A Deep Triplet Embedding for Fingerprint Liveness Detection

In this section, we describe the proposed method for fingerprint liveness detection based on triplet loss embedding. We start by describing the overall framework; subsequently, we introduce the network architecture and the training algorithm. Finally, we describe the matching procedure that leads to the final decision on the liveness of a given fingerprint image.

12.3.1 Framework

As depicted in Fig. 12.4, the proposed framework requires a collection of real and fake fingerprint images taken from a sensor and used as a training set. From each image, we randomly extract one fixed sized patch and arrange them in a certain number of triplets $\{x_i, x_j^+, x_k^-\}$, where x_i (anchor) and x_j^+ are two examples of the same class, and x_k^- comes from the other class. We alternatively set the anchor to be a real or a fake fingerprint patch.

The architecture is composed of three convolutional networks with shared weights so that three patches can be processed at the same time and mapped into a common feature space. We denote by $\mathbf{r}(\cdot)$ the representation of a given patch obtained from the output of one of the three networks. The deep features extracted from the live and fake fingerprints are compared in order to extract an intra-class distance $d(\mathbf{r}(x), \mathbf{r}(x^+))$ and an inter-class distance $d(\mathbf{r}(x), \mathbf{r}(x^-))$. The objective is to learn d so that the two examples of the same class are closer than two examples taken from different classes, and the distance between two examples of the same class is as short as possible. After training the networks with a certain number of triplets, we extract a new patch from each training sample and generate a new set of triplets. This procedure is repeated until convergence, see more details in Sect. 12.4.2.

After the training procedure is completed, the learned metric is used as a matching distance in order to establish the liveness of a new fingerprint image. Given a query fingerprint, we can extract p (possibly overlapping) patches and give them as input to

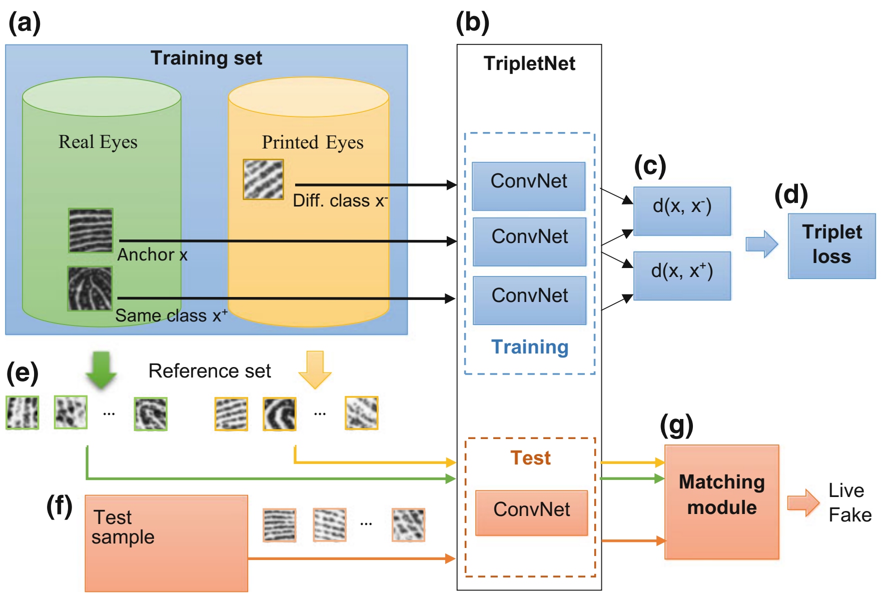

Fig. 12.4 The overall architecture of the proposed fingerprint liveness detection system. From the training set **a** of real and fake fingerprint acquisitions, we train a triplet network **b** using alternatively two patches of one class and one patch of the other one. The output of each input patch is used to compute the inter- and intra-class distances **c** in order to compute the objective function **d** that is used to train the parameters of the networks. After training, a set of real and a set of fake reference patches **e** are extracted from the training set (one for each fingerprint) and the corresponding representation is computed forwarding them through the trained networks. At test time, a set of patches is extracted from the fingerprint image **f** in order to map it to the same representation space as the reference gallery and are matched **g** in order to get a prediction on its liveness

the networks in order to get a representation $Q = \{\mathbf{r}(Q_1), \mathbf{r}(Q_2), \dots, \mathbf{r}(Q_p)\}$. Since we are not directly mapping each patch to a binary liveness label, but generating a more fine-grained representation, the prediction can be made by a decision rule based on the learned metric d computed with respect to a set R_L and R_F of real and fake reference fingerprints:

$$R_L = \{\mathbf{r}(x_{L_1}), \mathbf{r}(x_{L_2}), \dots, \mathbf{r}(x_{L_n})\} \tag{12.1a}$$

$$R_F = \{\mathbf{r}(x_{F_1}), \mathbf{r}(x_{F_2}), \dots, \mathbf{r}(x_{F_n})\} \tag{12.1b}$$

where the patches x_{L_i} and x_{F_i} can be taken from the training set or from a specially made design set.

Table 12.1 Architecture of the proposed embedding network

Layer name	Layer description	Output
input	32×32 gray level image	
conv1	5×5 conv. filters, stride 1, $1 \rightarrow 64$ feat. maps	$64 \times 28 \times 28$
batchnorm1	Batch normalization	$64 \times 28 \times 28$
relu1	Rectifier linear unit	$64 \times 28 \times 28$
conv2	3×3 conv. filters, stride 2, padding 1, $64 \rightarrow 64$ feat. maps	$64 \times 14 \times 14$
conv3	3×3 conv. filters, stride 1, $64 \rightarrow 128$ feat. maps	$64 \times 12 \times 12$
batchnorm2	Batch normalization	$64 \times 12 \times 12$
relu2	Rectifier linear unit	$64 \times 12 \times 12$
conv4	3×3 conv. filters, stride 2, padding 1, $128 \rightarrow 128$ feat. maps	$128 \times 6 \times 6$
conv5	3×3 conv. filters, stride 1, $128 \rightarrow 256$ feat. maps	$256 \times 4 \times 4$
batchnorm3	Batch normalization	$256 \times 4 \times 4$
relu3	Rectifier linear unit	$256 \times 4 \times 4$
conv6	3×3 conv. filters, stride 2, padding 1, $256 \rightarrow 256$ feat. maps	$256 \times 2 \times 2$
fc1	Fully connected layer $4 \times 256 \rightarrow 256$	256
dropout	Dropout $p = 0.4$	256
relu5	Rectifier linear unit	256
fc2	Fully connected layer $256 \rightarrow 256$	256
output	Softmax	256

12.3.2 Network Architecture

We employ a network architecture inspired by [37] where max pooling units, widely used for downsampling purposes, are replaced by simple convolution layers with increased stride. Table 12.1 contains the list of the operations performed by each layer of the embedding networks.

The architecture is composed of a first convolutional layer that takes the 32×32 grayscale fingerprint patches and outputs 64 feature maps by using filters of size 5×5. Then, batch normalization [38] is applied in order to get a faster training convergence and rectified linear units (ReLU) are used as nonlinearities. Another convolutional layer with a stride equal to 2, padding of 1 and filters of size 3×3 performs a downsampling operation by a factor of two in both directions.

The same structure is replicated two times, reducing the filter size to 3×3 and increasing the number of feature maps from 64 to 128 and from 128 to 256. At this point, the feature maps have the size of $128 \times 2 \times 2$ and are further processed by two fully connected layers with 256 outputs followed by a softmax layer. This nonlinearity helps in getting a better convergence of the training algorithm and ensures that the

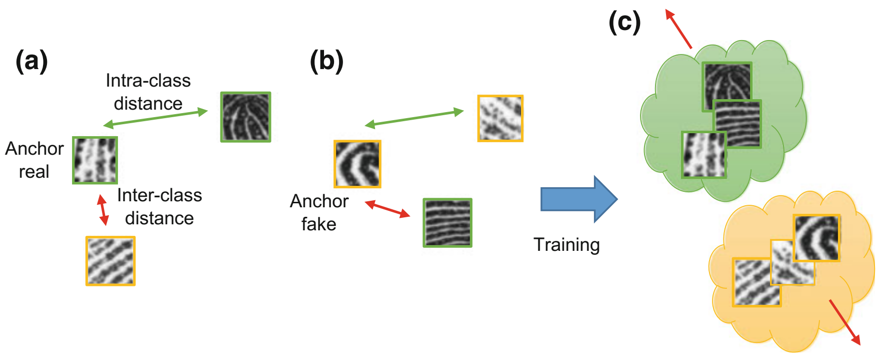

Fig. 12.5 The training procedure uses examples as triplets formed by **a** two real fingerprints (in *green*) and one impostor (in *yellow*) and **b** two impostors and one genuine. The training procedure using the triplet loss will result in an attraction for the fingerprints of the same class (either real or fake) so that their distance will be as close as possible. At the same time, real and fake fingerprints will be pushed away from each other (**c**)

distance among to outputs does not exceed one. Dropout [39] with probability 0.4 is applied to the first fully connected layer for regularization purposes.

The complete network is composed of three instances of this architecture: from three batches of fingerprint images we get the L2 distances between the matching and mismatching images. At test, we take the output of one of the three networks to obtain the representation for a given patch. If there are memory limitations, an alternative consists of using just one network, collapse the three batches into a single one, and computing the distances among the examples corresponding to the training triplets.

12.3.3 Training

As schematized in Fig. 12.5, the triplet architecture along with the triplet loss function aims to learn a metric that makes two patches of the same class closer with respect to two coming from different classes. The objective is to capture the cues that make two fingerprints both real or fake. The real ones come from different people and fingers, and their comparison is performed in order to find some characteristics that make them genuine. At the same time, fake fingerprints come from different people and can be built using several materials. The objective is to detect anomalies that characterize fingerprints coming from a fake replica, without regard to the material they are made of.

Given a set of triplets $\{x_i, x_j^+, x_k^-\}$, where x_i is the anchor and x_j^+ and x_k^- are two examples of the same and the other class, respectively, the objective of the original triplet loss [14] is to give a penalty if the following condition is violated:

$$d(\mathbf{r}(x_i), \mathbf{r}(x_j^+)) - d(\mathbf{r}(x_i), \mathbf{r}(x_k^-)) + 1 \leq 0 \tag{12.2}$$

At the same time, we would like to have the examples of the same class as close as possible so that, when matching a new fingerprint against the reference patches of the same class, the distance $d(\mathbf{r}(x_i), \mathbf{r}(x_j^+))$ is as low as possible. If we denote by $y(x_i)$ the class of a generic patch x_i, we can obtain the desired behavior by formulating the following loss function:

$$L = \sum_{i,j,k} \left\{ c(x_i, x_j^+, x_k^-) + \boldsymbol{\beta} c(x_i, x_j^+) \right\} + \lambda \|\boldsymbol{\theta}\|_2 \tag{12.3}$$

where θ is a one-dimensional vector containing all the trainable parameters of the network, $y(x_i) = y(x_j)$, $y(x_k^-) \neq y(x_i)$ and

$$c(x_i, x_j^+, x_k^-) = \max \left\{ 0, d(\mathbf{r}(x_i), \mathbf{r}(x_j^+)) - d(\mathbf{r}(x_i), \mathbf{r}(x_k^-)) + 1 \right\} \tag{12.4a}$$

$$c(x_i, x_j^+) = d(\mathbf{r}(x_i), \mathbf{r}(x_j^+)) \tag{12.4b}$$

During training, we compute the subgradients and use backpropagation through the network in order to get the desired representation. Contextualizing to what depicted in Fig. 12.5, $c(x_i, x_j^+, x_k^-)$ is the inter-class and $c(x_i, x_j^+)$ the intra-class distance term. $\lambda \|\theta\|_2$ is an additional weight decay term added to the loss function for regularization purposes.

After a certain number of iterations k, we periodically generate a new set of triplets by extracting a different patch from each training fingerprint. It is essential to not update the triplets after too many iterations because it can result in overfitting. At the same time, generating new triplets too often or mining hard examples can cause convergence issues.

12.3.4 Matching

In principle, any distance among bag of features can be used in order to match the query fingerprint $Q = \{\mathbf{r}(Q_1), \mathbf{r}(Q_2), \ldots, \mathbf{r}(Q_p)\}$ against the reference sets R_L and R_F. Since the training objective drastically pushes the distances to be very close to zero or to one, a decision on the liveness can be made by setting a simple threshold $\tau = 0.5$. An alternative could consists of measuring the Hausdorff distance between bags, but it would be too much sensitive to outliers since it involves the computation of the minimum distance between a test patch and each patch of each reference set. Even if using the k-th Hausdorff distance [40], that considers the k-th value instead of the minimum, we obtained better performance by following a simple majority voting strategy. It is also faster since it does not involve sorting out the distances.

Given a fingerprint Q, for each patch Q_j we count how many distances for each reference set are below the given threshold

$$D(R_L, Q_j) = |\{i \in \{1, \dots, n\} : d(R_{L_i}, Q_j) < \tau\}| \tag{12.5a}$$

$$D(R_F, Q_j) = |\{i \in \{1, \dots, n\} : d(R_{F_i}, Q_j) < \tau\}| \tag{12.5b}$$

then we make the decision evaluating how many patches belong to the real or the fake class:

$$y(Q) = \begin{cases} \text{real} & \text{if } \sum_{j=1}^{p} D(R_L, Q_j) \geq \sum_{j=1}^{p} D(R_F, Q_j) \\ \text{fake} & \text{otherwise} \end{cases} \tag{12.6}$$

The above method can also be applied in scenarios where multiple fingerprints are acquired from the same individual, as usually happens on passport checks at airports. For instance, the patches coming from different fingers can be accumulated in order to apply the same majority rule of Eq. 12.6 or the decision can be made on the most suspicious fingerprint.

12.4 Experiments

We evaluated the proposed approach with ten of the most popular benchmark for fingerprint liveness detection, coming from the LivDet competitions held in 2009 [7], 2011 [18] and 2013 [19]. We compare our method with the state of the art, specifically the VGG pretrained network of [9], the Local Contrast Phase Descriptor LCPD [29], the dense Scale Invariant Descriptor SID [30] and the Binarized Statistical Image Features [28]. For the main experiments, we strictly follow the competition rules using the training/test splits provided by the organizers while for the cross-dataset and cross-material scenarios, we follow the setup of [9].

The network architecture along with the overall framework have been implemented using the Torch7 computing framework [41] on an NVIDIA® DIGITS™ DevBox with four TITAN X GPUs with seven TFlops of single precision, 336.5 GB/s of memory bandwidth, and 12 GB of memory per board. MATLAB® has been used for image segmentation.

12.4.1 Datasets

The LivDet 2009 datasets [7] were released with the first international fingerprint liveness detection competition, with the aim of becoming a reference and allowing researchers to compare the performance of their algorithms or systems. The fingerprints were acquired using the cooperative approach (see Sect. 12.2.1) and the replicas are created using the materials: gelatin, silicone, and play-doh. The organizers released three datasets, acquired using three different sensors: Biometrika (FX2000), Identix (DFR2100), and Crossmatch (Verifier 300 LC).

Table 12.2 Details of the LivDet 2009 and 2013 competitions. The last row indicates the spoof materials: S = Silicone, G = Gelatine, P = Play-Doh, E = Ecoflex, L = Latex, M = Modasil, B = Body Double, W = Wooden glue

Competition	LivDet2009			LivDet2013		
Dataset	Biometrika	CrossMatch	Identix	Biometrika	Italdata	Swipe
Size	312 × 372	480 × 640	720 × 720	312 × 372	480 × 640	1500 × 208
DPI	569	500	686	569	500	96
Subjects	50	254	160	50	50	100
Live samples	2000	2000	1500	2000	2000	2500
Spoof samples	2000	2000	1500	2000	2000	2000
Materials	S	GPS	GPS	EGLMW	EGLMW	BLPW

The LivDet 2011 competition [18] released four datasets, acquired using the scanners Biometrika (FX2000), Digital Persona (4000B), ItalData (ETT10) and Sagem (MSO300). The materials used for fake fingerprints are gelatin, latex, Ecoflex (platinum-catalyzed silicone), silicone and wooden glue. The spoof fingerprints have been obtained as in LivDet 2009 with the cooperative method.

The LivDet 2013 competition [19] consists of four datasets acquired using the scanners Biometrika (FX2000), ItalData (ETT10), Crossmatch (L SCAN GUARDIAN) and Swipe. Differently from LivDet 2011, two datasets, Biometrika and Italdata, have been acquired using the non-cooperative method. That is, latent fingerprints have been acquired from a surface, and then printed on a circuit board (PCB) in order to generate a three-dimensional structure of the fingerprint that can be used to build a mold. To replicate the fingerprints they used Body Double, latex, PlayDoh and wood glue for the Crossmatch and Swipe datasets and gelatin, latex, Ecoflex, Modasil and wood glue for Biometrika and Italdata.

The size of the images, the scanner resolution, the number of acquired subject and of live and fake samples are detailed in Tables 12.2 and 12.3. The partition of training and test examples is provided by the organizers of the competition.

12.4.2 Experimental Setup

For all the experiments we evaluate performance in terms of average classification error. This is the measure used to evaluate the entries in the LivDet competitions and is the average of the Spoof False Positive Rate (SFPR) and the Spoof False Negative Rate (SFNR). For all the experiments on the LivDet test sets we follow the standard protocol and since a validation set is not provided, we reserved a fixed amount of 120 fingerprints. For the cross-dataset experiments, we used for validation purposes the Biometrika 2009 and Crossmatch 2013 datasets.

Table 12.3 Details of the LivDet 2011 competition. The last row indicates the spoof materials: Sg = Silgum, the others are the same as in Table 12.2

Competition	LivDet2011			
Dataset	Biometrika	Digital persona	Italdata	Sagem
Size	312 × 372	355 × 391	640 × 480	352 × 384
DPI	500	500	500	500
Subjects	200	82	92	200
Live samples	2000	2000	2000	2000
Spoof samples	2000	2000	2000	2000
Materials	EGLSgW	GLPSW	EGLSgW	GLPSW

The triplets set for training is generated by taking one patch from each fingerprint and arranging them alternatively in two examples of one class and one of the other class. The set is updated every $k = 100{,}000$ triplets that are fed to the networks in batches of 100. In the remainder of the chapter, we refer to each update as the start of a new iteration. We use stochastic gradient descent to minimize the triplet loss function, setting a learning rate of 0.5 and a momentum of 0.9. The learning rate η_0 is annealed by following the form:

$$\eta = \frac{\eta_0}{1 + 10^{-4} \cdot b} \tag{12.7}$$

where b is the progressive number of batches that are being processed. That is, after ten iterations the learning rate is reduced by half. The weight decay term of Eq. 12.3 is set to $\lambda = 10^{-4}$ and $\beta = 0.002$ as in [17].

After each iteration, we check the validation error. Instead of using the same accuracy measured at test (the average classification error), we construct 100,000 triplets using the validation set patches, but taking as anchor the reference patches taken from the training set and used to match the test samples. The error consists of the number of violating triplets and reflects how much the reference patches failed to classify patches never seen before. Instead of fixing the number of iterations, we employ early stopping based on the concept of patience [42]. Each time the validation error decrease, we save a snapshot of the network parameters, and if in 20 consecutive iterations the validation error is not decreasing anymore, we stop the training and evaluate the accuracy on the test set using the last saved snapshot.

12.4.3 Preprocessing

Since the images coming from the scanners contain a wide background area surrounding the fingerprint, we segmented the images in order to avoid extracting background patches. The performance is highly affected by the quality of the background subtraction, therefore, we employed an algorithm [43], that divides the fingerprint image into 16×16 blocks, and classifies a block as foreground only if its standard deviation is more than a given threshold. The rationale is that a higher standard deviation corresponds to the ridge regions of a fingerprint. In order to exclude background noise that can interfere with the segmentation, we compute the connected components of the foreground mask and take the fingerprint region as the one with the largest area. In order to get a smooth segmentation, we generate the convex hull image from the binary mask using morphological operations.

We also tried to employ data augmentation techniques in terms of random rotations, flipping and general affine transformation. Anyway, they significantly slowed down the training procedure and we did not get any performance improvement on either the main and cross-dataset experiments.

12.4.4 Experimental Results

In this section, we present the performance of the proposed fingerprint liveness detection system in different scenarios. In Table 12.4 we list the performance in terms of average classification error on the LivDet competition test sets. With respect to the currently best-performing methods [9, 29, 30] we obtained competitive performance for all the datasets, especially on Italdata 2011, and Swipe 2013. This means that the approach works properly also on the images coming from swipe scanners, where the fingerprints are acquired by swiping the finger across the sensor surface (see Fig. 12.3e, f). Overall, our approach has an average error of 1.75% in comparison

Table 12.4 Average classification error for the LivDet Test Datasets. In column 2 our TripletNet based approach, in column 2 the VGG deep network pretrained on the Imagenet dataset and fine-tuned by [9], in column 3 the Local Contrast Phase Descriptor [29] based approach, in column 4 the dense Scale Invariant Descriptor [30] based approach and in column 5 the Binarized Statistical Image Features [28] based approach

Dataset	TripletNet	VGG [9]	LCPD [29]	SID [30]	BSIF [28]
Biometrika 2009	**0.71**	4.1	1	3.8	9
CrossMatch 2009	1.57	**0.6**	3.4	3.3	5.8
Identix 2009	**0.044**	0.2	1.3	0.7	0.7
Biometrika 2011	5.15	5.2	**4.9**	5.8	6.8
Digital 2011	**1.85**	3.2	4.7	2.0	4.1
Italdata 2011	**5.1**	8	12.3	11.2	13.9
Sagem 2011	**1.23**	1.7	3.2	4.2	5.6
Biometrika 2013	**0.65**	1.8	1.2	2.5	1.1
Italdata 2013	0.5	**0.4**	1.3	2.7	3
Swipe 2013	**0.66**	3.7	4.7	9.3	5.2
Average	**1.75%**	2.89%	3.8%	4.5%	5.5%

to the 2.89% of [9] which results in a performance improvement of 65%. We point out that we did not use the dataset CrossMatch 2013 for evaluation purposes because the organizers of the competition found anomalies in the data and discouraged its use for comparative evaluations [4]. In Fig. 12.6 we depict a 2D representation of the test set of Biometrika 2013, specifically one patch for every fingerprint image, computed from an application of t-SNE [44] to the generated embedding. This dimensionality reduction technique is particularly insightful since it maps the high-dimensional representation in a space where the vicinity of points is preserved. We can see that the real and fake fingerprints are well separated and only a few samples are in the wrong place, for the major part Wooden Glue and Modasil. Ecoflex and gelatin replicas seem more easy to reject. Examining the patch images, we can see that going top to bottom, the quality of the fingerprint pattern degrades. This may be due to the perspiration of the fingertips that makes the ridges not as uniform as the fake replicas.

12.4.4.1 Cross-Dataset Evaluation

As in [9], we present some cross-dataset evaluation and directly compare our performance with respect to their deep learning and Local Binary Pattern approach. The results are shown in Table 12.5 and reflect a significant drop in performance with respect to the previous experiments. With respect to [9] the average classification error is slightly better, anyway it is too high to possibly consider doing liveness detection in the wild. Similar results have been obtained by [34]. We point out that different sensors, settings and climatic conditions can extremely alter the fingerprint images, and if the training set is not representative of the particular conditions, any machine learning approach, not just deep learning algorithms, would not be effective at generalization.

12.4.4.2 Cross-Material Evaluation

We also evaluated the robustness of our system to new spoofing materials. We followed the protocol of [9] by training the networks using a subset of materials and testing on the remaining ones. The results are shown in Table 12.6. With respect to the cross-dataset experiments, the method appears to be more robust to new materials rather than a change of the sensor. Also in this scenario, if we exclude the Biometrika 2011 dataset, our approach has a significative improvement with respect to [9].

12.4.4.3 Computational Efficiency

One of the main benefits of our approach is the computational time since the architecture we employed is smaller in comparison to other deep learning approaches such as [9, 34]. Moreover, the patch representation allows for scaling the matching

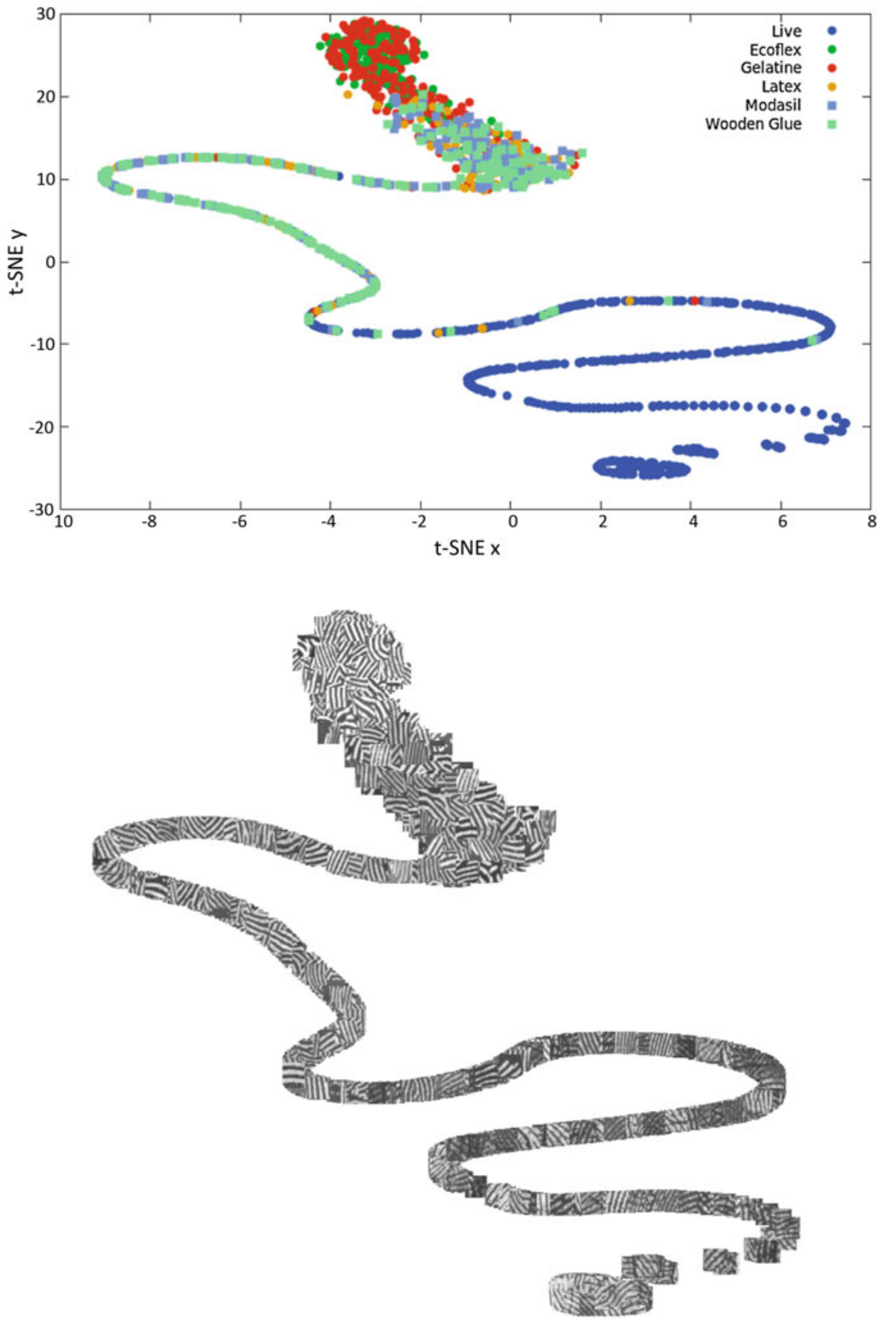

Fig. 12.6 T-SNE visualization of the embedding generated from the live and fake fingerprints composing the test set of Biometrika 2013 (one patch for each acquisition). The high dimensional representation is mapped into a two-dimensional scatter plot where the vicinity of points is preserved

Table 12.5 Average classification error for the cross-dataset scenarios. The first column is the dataset used for training and the second the one used for the test. The third column is our TripletNet approach, the fourth and the fifth are the deep learning and the Local Binary Patterns (LBP) based approaches proposed by [9]

Training set	Test set	TripletNet	VGG	LBP
Biometrika 2011	Biometrika 2013	14	15.5	16.5
Biometrika 2013	Biometrika 2011	34.05	46.8	47.9
Italdata 2011	Italdata 2013	8.3	14.6	10.6
Italdata 2013	Italdata 2011	44.65	46.0	50.0
Biometrika 2011	Italdata 2011	29.35	37.2	47.1
Italdata 2011	Biometrika 2011	27.65	31.0	49.4
Biometrika 2013	Italdata 2013	1.55	8.8	43.7
Italdata 2013	Biometrika 2013	3.8	2.3	48.4

Table 12.6 Average Classification Error for the cross-material scenario. In column 2 are the materials used for training and in column 3 the ones used for the test. The abbreviations are the same as in Tables 12.2 and 12.3

Dataset	Train materials	Test materials	TripletNet	VGG	LBP
Biometrika 2011	EGL	SgW	13.1	10.1	17.7
Biometrika 2013	MW	EGL	2.1	4.9	8.5
Italdata 2011	EGL	SgW	7	22.1	30.9
Italdata 2013	MW	EGL	1.25	6.3	10.7

procedure on different computational units, so that it can be used also in heavily populated environments. In our experiments, we extract 100 patches from each test fingerprint and the time to get their corresponding representation is about 0.6ms using a single GPU and 0.3 s using a Core i7-5930K 6 Core 3.5 GHz desktop processor (single thread). Considering the most common dataset configuration of 880 real and 880 fake reference patches, the matching procedure takes 5.2 ms on a single GPU and 14 ms on the CPU. Finally, the training time varies depending on the particular dataset, and on the average, the procedure converges in 135 iterations. A single iteration takes 84 and 20 s are needed to check the validation error.

12.5 Conclusions

In this chapter, we introduced a novel framework for fingerprint liveness detection which embeds the recent advancements in deep metric learning. We validated the effectiveness of our approach in a scenario where the fingerprints are acquired using the same sensing devices that are used for training. We also presented quantified results on the generalization capability of the proposed approach for new acquisition devices, and unseen spoofing materials. The approach is able to work in real time and surpasses the state-of-the-art on several benchmark datasets.

In conclusion, we point out that the employment of software-based liveness detection systems should never give a sense of false security to their users. As in other areas such as cyber-security, the attackers become more resourceful every day and new ways to fool a biometric system will be discovered. Therefore, such systems should be constantly updated and monitored, especially in critical applications such as airport controls. It would be desirable to have large datasets that contain fingerprint images of people with different age, sex, ethnicity, and skin conditions and that are acquired under different time periods, environments and using a variety of sensors with a multitude of spoofing materials.

Acknowledgements This work was supported in part by grant N00014-12-1-1026.

References

1. A.K. Jain, A. Ross, S. Pankanti, Biometrics: a tool for information security. IEEE Trans. Inf. Forensics Secur. **1**(2), 125–143 (2006)
2. M. Martinez-Diaz, J. Fierrez, J. Galbally, J. Ortega-Garcia, An evaluation of indirect attacks and countermeasures in fingerprint verification systems. Pattern Recogn. Lett. **32**(12), 1643–1651 (2011)
3. T. Matsumoto, H. Matsumoto, K. Yamada, S. Hoshino, Impact of artificial “gummy” fingers on fingerprint systems. Proc. Conf. Opt. Secur. Counterfeit Deterrence Tech. **4677**, 275–289 (2002)
4. L. Ghiani, D.A. Yambay, V. Mura, G.L. Marcialis, F. Roli, S.A. Schuckers, Review of the fingerprint liveness detection (livdet) competition series: 2009 to 2015. *Image and Vision Computing* (2016) (in press)
5. C. Sousedik, C. Busch, Presentation attack detection methods for fingerprint recognition systems: a survey. IET Biom. **3**(4), 219–233 (2014)
6. J. Galbally, S. Marcel, J. Fierrez, Biometric antispoofing methods: a survey in face recognition. IEEE Access **2**, 1530–1552 (2014)
7. G.L. Marcialis, A. Lewicke, B. Tan, P. Coli, D. Grimberg, A. Congiu, A. Tidu, F. Roli, S. Schuckers, First international fingerprint liveness detection competition – livdet 2009, in *Proceedings of the International Conference on Image Analysis and Processing (ICIAP, 2009)* (Springer, Berlin, 2009), pp. 12–23
8. D. Menotti, G. Chiachia, A. Pinto, W.R. Schwartz, H. Pedrini, A.X. Falcao, A. Rocha, Deep representations for iris, face, and fingerprint spoofing detection. IEEE Trans. Inf. Forensics Secur. **10**(4), 864–879 (2015)
9. R.F. Nogueira, R. de Alencar Lotufo, R.C. Machado, Fingerprint liveness detection using convolutional neural networks. IEEE Trans. Inf. Forensics Secur. **11**(6), 1206–1213 (2016)
10. S. Kim, B. Park, B.S. Song, S. Yang, Deep belief network based statistical feature learning for fingerprint liveness detection. Pattern Recogn. Lett. **77**, 58–65 (2016)
11. A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in *Conference on Advances in Neural Information Processing Systems (NIPS, 2012)* (2012), pp. 1097–1105
12. Y. LeCun, Y. Bengio, G. Hinton, Deep learning. Nature **521**(7553), 436–444 (2015)
13. I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning* (MIT Press, Cambridge, 2016), http://www.deeplearningbook.org

14. E. Hoffer, N. Ailon, Deep metric learning using triplet network, in *Proceedings of the International Workshop on Similarity-Based Pattern Recognition (SIMBAD, 2015)* (Springer, Berlin, 2015), pp. 84–92
15. J. Bromley, J.W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, R. Shah, Signature verification using a "siamese" time delay neural network. Int. J. Pattern Recognit. Artif. Intell. **7**(04), 669–688 (1993)
16. P. Wohlhart, V. Lepetit, Learning descriptors for object recognition and 3D pose estimation, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR, 2015)* (2015), pp. 3109–3118
17. D. Cheng, Y. Gong, S. Zhou, J. Wang, N. Zheng, Person re-identification by multi-channel parts-based cnn with improved triplet loss function, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR, 2016)* (2016)
18. D. Yambay, L. Ghiani, P. Denti, G.L. Marcialis, F. Roli, S. Schuckers, Livdet 2011 - fingerprint liveness detection competition 2011, in *Proceedings of the IAPR International Conference on Biometrics (ICB, 2011)* (2012), pp. 208–215
19. L. Ghiani, D. Yambay, V. Mura, S. Tocco, G.L. Marcialis, F. Roli, S. Schuckcrs, Livdet 2013 fingerprint liveness detection competition 2013, in *Proceedings of the International Conference on Biometrics (ICB, 2013)* (2013), pp. 1–6
20. A. Kumar, Y. Zhou, Human identification using finger images. IEEE Trans. Image Process. **21**(4), 2228–2244 (2012)
21. K. Seifried, Biometrics-what you need to know. Secur. Portal **10** (2001)
22. D. Baldisserra, A. Franco, D. Maio, D. Maltoni, Fake fingerprint detection by odor analysis, in *Proceedings of the International Conference on Biometrics (ICB, 2006)* (Springer, Berlin, 2006), pp. 265–272
23. P.V. Reddy, A. Kumar, S.M.K. Rahman, T.S. Mundra, A new antispoofing approach for biometric devices. IEEE Trans. Biomed. Circuits Syst. **2**(4), 328–337 (2008)
24. A. Antonelli, R. Cappelli, D. Maio, D. Maltoni, Fake finger detection by skin distortion analysis. IEEE Trans. Inf. Forensics Secur. **1**(3), 360–373 (2006)
25. R. Derakhshani, S.A.C. Schuckers, L.A. Hornak, L. O'Gorman, Determination of vitality from a non-invasive biomedical measurement for use in fingerprint scanners. Pattern Recogn. **36**(2), 383–396 (2003)
26. B. Tan, S. Schuckers, Liveness detection for fingerprint scanners based on the statistics of wavelet signal processing, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshop (CVPRW, 2006)* (IEEE, 2006), p. 26
27. S.B. Nikam, S. Agarwal, Texture and wavelet-based spoof fingerprint detection for fingerprint biometric systems, in *Proceedings of the International Conference on Emerging Trends in Engineering and Technology (ICETET, 2008)* (IEEE, 2008), pp. 675–680
28. L. Ghiani, A. Hadid, G.L. Marcialis, F. Roli, Fingerprint liveness detection using binarized statistical image features, in *Proceedings of the International Conference on Biometrics Theory, Applications and Systems (BTAS, 2013)* (IEEE, 2013), pp. 1–6
29. D. Gragnaniello, G. Poggi, C. Sansone, L. Verdoliva, Local contrast phase descriptor for fingerprint liveness detection. Pattern Recogn. **48**(4), 1050–1058 (2015)
30. D. Gragnaniello, G. Poggi, C. Sansone, L. Verdoliva, An investigation of local descriptors for biometric spoofing detection. IEEE Trans. Inf. Forensics Secur. **10**(4), 849–863 (2015)
31. I. Kokkinos, M. Bronstein, A. Yuille, Dense Scale Invariant Descriptors for Images and Surfaces. Research report RR-7914, INRIA (2012)
32. K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition (2014), arXiv.org/abs/1409.1556
33. R. Olga, D. Jia, S. Hao, K. Jonathan, S. Sanjeev, M. Sean, H. Zhiheng, K. Andrej, K. Aditya, B. Michael, C.B. Alexander, F. Li, Imagenet large scale visual recognition challenge. Int. J. Comput. Vision **115**(3), 211–252 (2015)
34. E. Marasco, P. Wild, B. Cukic, Robust and interoperable fingerprint spoof detection via convolutional neural networks (hst 2016), in *Proceedings of the IEEE Symposium on Technologies for Homeland Security* (2016), pp. 1–6

35. C. Wang, K. Li, Z. Wu, Q. Zhao, *A DCNN Based Fingerprint Liveness Detection Algorithm with Voting Strategy* (Springer International Publishing, Cham, 2015). pp. 241–249
36. G.E. Hinton, Training products of experts by minimizing contrastive divergence. Neural Comput. **14**(8), 1771–1800 (2002)
37. J.T. Springenberg, A. Dosovitskiy, T. Brox, M. Riedmiller, Striving for simplicity: the all convolutional net (2014), arXiv.org/abs/1412.6806
38. S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, in *Proceedings of the International Conference on Machine Learning (ICML, 2015)* (2015), p. 37
39. N. Srivastava, G.E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
40. J. Wang, J. Zucker, Solving multiple-instance problem: a lazy learning approach (2000), pp. 1119–1126
41. R. Collobert, K. Kavukcuoglu, C. Farabet, Torch7: a matlab-like environment for machine learning, in *BigLearn, NIPS Workshop* (2011)
42. Y. Bengio, Practical recommendations for gradient-based training of deep architectures, in *Neural Networks: Tricks of the Trade* (Springer, Berlin, 2012), pp. 437–478
43. P.D. Kovesi, MATLAB and Octave functions for computer vision and image processing (2005), http://www.peterkovesi.com/matlabfns
44. L. van der Maaten, G. Hinton, Visualizing data using t-sne. J. Mach. Learn. Res. **9**, 2579–2605 (2008)