Chapter 8
Presentation Attack Detection for Face in Mobile Phones

Yaojie Liu, Joel Stehouwer, Amin Jourabloo, Yousef Atoum and Xiaoming Liu

Abstract Face is the most accessible biometric modality which can be used for identity verification in mobile phone applications, and it is vulnerable to many different presentation attacks, such as using a printed face/digital screen face to access the mobile phone. Presentation attack detection is a very critical step before feeding the face image to face recognition systems. In this chapter, we introduce a novel two-stream CNN-based approach for the presentation attack detection, by extracting the patch-based features and holistic depth maps from the face images. We also introduce a two-stream CNN v2 with model optimization, compression and a strategy of continuous updating. The CNN v2 shows great performances of both generalization and efficiency. Extensive experiments are conducted on the challenging databases (CASIA-FASD, MSU-USSA, replay attack, OULU-NPU, and SiW), with comparison to the state of the art.

8.1 Introduction

Biometrics authentication systems aim to utilize physiological characteristics, such as fingerprint, face, and iris, or behavioral characteristics, such as typing rhythm and gait, to uniquely identify an individual. As biometric systems are widely used in real-world applications including unlocking cell phone and granting mobile

Y. Liu · J. Stehouwer · A. Jourabloo · Y. Atoum · X. Liu (✉)
Department of Computer Science and Engineering,
Michigan State University, East Lansing, MI 48824, USA
e-mail: liuxm@msu.edu

Y. Liu
e-mail: liuyaoj1@msu.edu

J. Stehouwer
e-mail: stehouw7@msu.edu

A. Jourabloo
e-mail: jourablo@msu.edu

Y. Atoum
e-mail: atoumyou@msu.edu

A. Rattani et al. (eds.), *Selfie Biometrics*, Advances in Computer Vision and Pattern Recognition, https://doi.org/10.1007/978-3-030-26972-2_8

transaction, biometric spoofs, or presentation attacks (PA) are becoming a large threat, where a spoof biometric sample is presented to the biometric system and attempts to be authenticated. Face, as the most accessible biometric modality, has many different types of PAs including print attack, replay attack, 3D masks, etc. As a result, conventional face recognition systems can be very vulnerable to such PAs and are exposed to risk and loss beyond measure.

In order to develop a face recognition system that is invulnerable to various types of PAs, there is an increasing demand in designing a robust presentation attack detection (PAD or face anti-spoofing) system to classify a face sample as live/spoof *before* recognizing its identity. Previous approaches to PAD can be categorized into three groups. The first is the texture-based methods, which discover discriminative texture characteristics unique to various attack mediums. Due to a lack of an explicit correlation between pixel intensities and different types of attacks, extracting robust texture features is challenging. The second is the motion-based methods that aim at classifying face videos based on detecting movements of facial parts, e.g., eye blinking and lip movements. These methods are suitable for static attacks, but not dynamic attacks such as replay or mask attacks. The third is image quality and reflectance-based methods, which design features to capture the superimposed illumination and noise information to the spoof images.

Most of the prior face PAD works apply SVM on hand-crafted features. While convolutional neural network (CNN) exhibits a superior performance in many computer vision tasks [6, 31, 32], there are only a few CNN-based methods for face PAD. Those methods typically use CNN for learning the representations, which will be further classified by SVM [33, 42]. In our view, further utilizing CNN in multiple ways, such as end-to-end training and learning with additional supervision, is a viable option for solving face PAD problems. On the one hand, with an increasing variety of sensing environments and PAs, it is not desirable to have a hand-crafted feature to cover all attacks. On the other hand, we need CNN to learn a robust feature from the data. With the growing numbers of face spoofing databases, CNN is known to be able to leverage the larger amount of training data and learn generalizable information to discriminate live versus spoof samples.

Following this perspective, in this chapter, we introduce a novel two-stream CNN-based face PAD method for print and replay attacks, denoted as CNN v1. The proposed method extracts the patch-based features and holistic depth maps from face images, as shown in Fig. 8.1. Here, the patch-based features are extracted from a local region of the face images, aiming at learning the spoofing texture that exists all over the images. The depth map leverages the whole face and describes the live face as a 3D object but the printed and digital screen face as a flat plain. Combining the patch-based and holistic features has two benefits: First, utilizing the local patches help to learn spoof patterns independent of spatial face areas. Second, holistic depth maps leverage the physical properties of the spoof attacks and learn a pixel-wise labeling. We use two CNNs to learn patch-based and holistic features, respectively. The first CNN is trained to predict a score for each extracted patch from a face image, and we assign the face image with the average of scores. The second CNN estimates the depth map of the face image and provides the face image with a liveness score

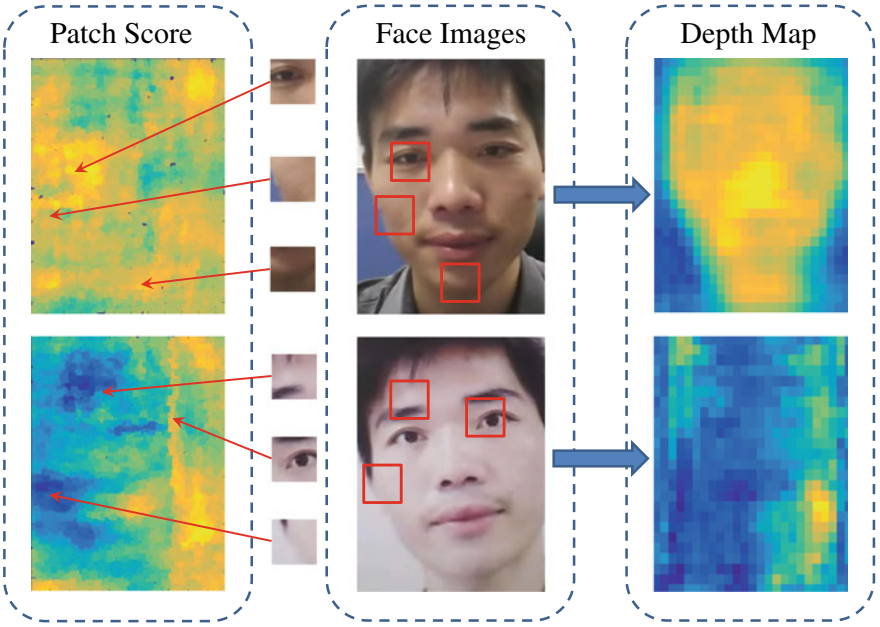

Fig. 8.1 In order to differentiate between live from spoof images, we propose an approach fusing patch-based and holistic depth-based cues. Left column shows the output scores of the local patches for a live image (top) and a spoof image (bottom), where the blue/yellow represents a high/low probability of spoof. While this visualization utilizes densely sampled patches, 10 random patches are sufficient for our anti-spoof classification. Right column shows the output of holistic depth estimation, where the yellow/blue represents closer/further points

based on estimated depth map. The fusion of the scores of both parts leads to the final estimated class of live versus spoof. The combination of these patch-based and depth-based CNNs is referred to as CNN v1. Further, to embed such PAD method in a mobile scenario, we apply an architecture optimization, a model compression, and a strategy of continuous updating. We call the advanced model as the two-stream CNN v2. The CNN v2 is trained in an end-to-end fashion and obtains comparable or higher accuracy in comparison with CNN v1, while achieving a real-time efficiency on the mobile phone system.

We summarize the contributions of this chapter as follows:

- Our proposed method utilizes both learned local and holistic features for classifying live versus spoof face samples;
- We propose a method for estimating the dense depth map for a live or spoof face image;
- We achieve the state-of-the-art performance on conventional face anti-spoofing databases;
- We provide an practical approach to train a robust and efficient system for mobile PAD scenarios.

8.2 Prior Work

We review papers in three relevant areas: traditional face PAD methods, CNN-based PAD methods, and image depth estimation.

Traditional face PAD methods Most prior work utilizes hand-crafted features and adopts shallow learning techniques (e.g., SVM and LDA) to develop a PAD system. A great number of works pay attention to the texture differences between the live faces and the spoof ones. Common local features that have been used in prior work include LBP [18, 19, 38], HOG [30, 58], DoG [44, 53], SIFT [41], and SURF [8]. However, the aforementioned features to detect texture difference could be very sensitive to different illuminations, camera devices, and specific identities. Researchers also seek solutions on different color spaces such as HSV and YCbCr [7, 10], Fourier spectra [37], and optical flow maps (OFM) [4].

Additionally, some approaches attempt to leverage the spontaneous face motions. Eye blinking is one cue proposed in [40, 52], to detect spoof attacks such as paper attack. In [29], Kollreider et al. use lip motion to monitor the face liveness. Methods proposed in [14, 15] combine audio and visual cues to verify the face liveness.

CNN-based methods CNNs have been proven to successfully outperform other learning paradigms in many computer vision tasks [6, 31, 32]. In [33, 42], the CNN serves as a feature extractor. Both methods fine-tune their network from a pre-trained model (CaffeNet in [42], VGG-face model in [33]) and extract the features to distinguish live versus spoof. In [59], Yang et al. propose to learn a CNN as a classifier for face PAD. Registered face images with different spatial scales are stacked as input, and live/spoof labeling is assigned as the output. In addition, Feng et al. [20] propose to use multiple cues as the CNN input for live/spoof classification. They select shearlet-based features to measure the image quality and the OFM of the face area as well as the whole scene area. And in [57], Xu et al. propose an LSTM-CNN architecture to conduct a joint prediction for multiple frames of a video.

However, compared to other face-related problems, such as face recognition [32, 36, 55] and face alignment [26], there are still substantially fewer efforts and exploration on face PAD using deep learning techniques [3, 27, 34]. Therefore, the proposed method aims to further explore the capability of CNN in face PAD, from the novel perspective of fusing the local texture-based decision and holistic depth maps.

Image depth estimation Estimating depth from a single RGB image is a fundamental problem in computer vision. In recent years, there has been rapid progress due to data-driven methods [28], especially deep neural networks trained on large RGB-D datasets [50], as well as weak annotations [12]. Specifically, for face images, face reconstruction from one image [24, 26, 54] or multiple images [46, 47] can also be viewed as one approach for depth estimation. However, to the best of our knowledge, no prior work has attempted to estimate the depth for a spoof image, such as a face on a printed paper. In contrast, our approach estimates depth for both the live face and spoof face, which is particularly challenging since the CNN needs to discern the subtle difference between two cases in order to correctly infer the depth.

8.3 Robust CNN System for Mobile PAD

In this section, we present the details of the proposed CNN system for Mobile PAD. We first introduce a general CNN system denoted as CNN v1, which leverages two streams of CNNs: patch-based CNN and depth-based CNN. To tailor the system for a mobile scenario, we redesign the patch-based CNN and combine it with the depth-based CNN, denoted as CNN v2. In addition, we propose a simple but effective learning strategy of continuous updating to improve the robustness of the system.

8.3.1 Patch- and Depth-Based CNN v1

The proposed CNN v1 [3] consists of two streams: patch-based CNN and depth-based CNN. Figure 8.2 shows a high-level illustration of both streams along with a fusion strategy for combining them. For the patch-based CNN stream, we train a deep neural network end-to-end to learn rich appearance features, which are capable of discriminating between live and spoof face images using patches randomly extracted from face images. For the depth-based CNN stream, we train a fully convolutional network (FCN) to estimate the depth of a face image, by assuming that a print or replay presentation attack has a flat depth map, while live faces contain a normal face depth.

Either the appearance or the depth cue can detect face attacks independently. However, fusing both cues has proven to provide promising results. In this model, we refer to the fusion output as the spoof score. A face image or video clip is classified as spoof if its spoof score is above a pre-defined threshold. In the remainder of this section, we explain in detail the two CNN steams used for face PAD.

8.3.1.1 Patch-Based CNN

There are multiple motivations to use patches instead of full face in our CNN. First is to increase the number of training samples for CNN learning. Note that for all available anti-spoofing datasets, only a limited number of samples are available for

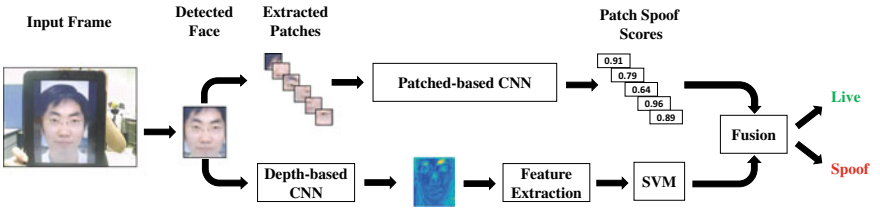

Fig. 8.2 Architecture of the proposed face PAD approach

training. For example, CASIA-FASD only contains 20 training subjects, with 12 videos per subject. Even though hundreds of faces can be extracted from each video, overfitting could be a major issue when learning the CNN due to the high similarities across the frames. Second, when using the full face images as input, traditional CNN needs to resize faces due to varying face image resolutions, where such scaling change might lead to the reduction of the discriminative information. In contrast, using the local patches can maintain the native resolution of the original face images, and thus preserve the discriminative ability. Third, assuming the spoof-specific discriminative information is present spatially in the entire face region, and patch-level input can enforce CNN to discover such information, regardless of the patch location. This is a more constrained or challenging learning task compared to using the whole face image.

Input features CNN is claimed to be a powerful feature learner that is able to map from raw *RGB* pixel intensities to the discriminative feature representation, guided by the loss function, which is in sharp difference to the conventional hand-crafted features. In our work, one observation is that CNN might also benefit from the hand-crafted features, which are proven to work well for the anti-spoof application. In a way, this is one form of bringing domain knowledge to CNN learning. This might be especially important for face anti-spoof applications, since without domain knowledge it is more likely for CNN to learn non-generalizable information from the data, rather than the true discriminative feature.

In reviewing hand-crafted features for face PAD, researchers have been experimenting with several color spaces as input to a feature extraction module to find discriminative descriptors. Typically, the most common color spaces used are *RGB*, *HSV*, YC_bC_r, and several combinations among them, such as $HSV + YC_bC_r$ [10]. The *RGB* has limited applications in face PAD due to the high correlation between the three color components and the imperfect separation of the luminance and chrominance information. On the other hand, *HSV* and YC_bC_r are based on the separation of the luminance and the chrominance information, providing additional features for learning the discriminative cues.

In this work, we attempt to use both *HSV* and YC_bC_r color spaces in the CNN-based methods. Moreover, we also explore several other input feature maps to the CNN including a pixel-wise *LBP* map and high-frequency patches. For the pixel-wise *LBP* map, we use the $LBP_{8,1}$ operator (i.e., $P = 8$ and $R = 1$) to extract the pixel-wise textural features from the face image, and afterward we randomly extract patches from the texture map. Note that in previous works, *LBP* is only used to extract histogram descriptors. For the high-frequency patches, the idea is to remove the low-frequency information from the patches which is motivated by the work in [17]. For any given face image $\mathbf{I}$, we subtract the low-pass filtered image of $\mathbf{I}$, which results in a high-frequency image $\mathbf{I}_H = \mathbf{I} - f_{lp}(\mathbf{I})$. An illustration of the various input features explored in our system is in Fig. 8.3. Compared to using RGB alone, providing these input features can facilitate the CNN training.

Based on our experiments, all of the proposed input features are useful representations to learn a CNN capable of distinguishing spoof attacks from live faces. In the experiments section, quantitative results comparing the input features will be

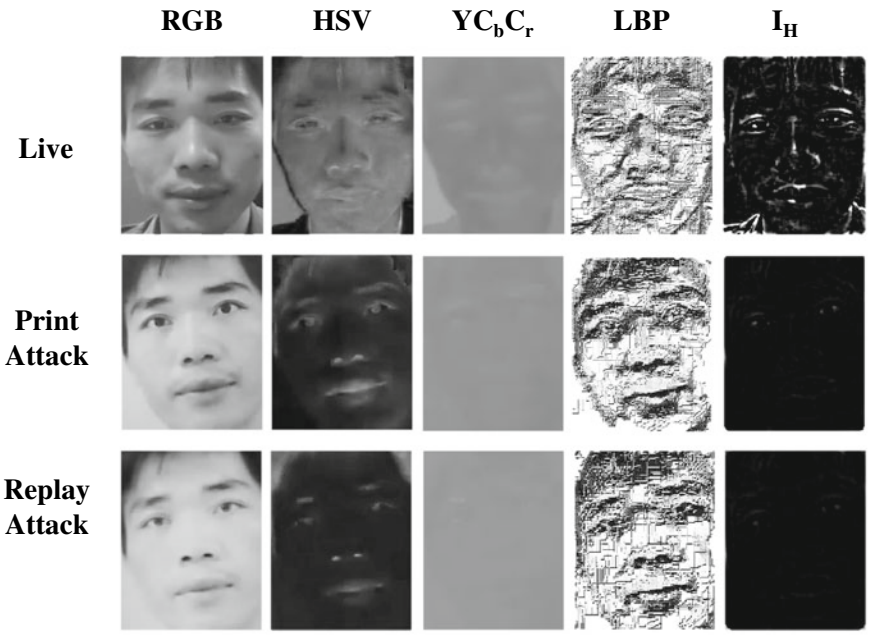

Fig. 8.3 Examples on *RGB* (G channel), *HSV* (S channel), YC_bC_r (C_b channel), pixel-wise *LBP* (*LBP* of S channel in *HSV*), high-frequency images (using G in *RGB*) of both live and spoof face images

presented. For the patch-based CNN, after detecting the face region, we convert the full face image into one of the feature representations, i.e., *HSV*, and then extract fixed size patches for CNN training and testing.

8.3.1.2 Depth-Based CNN

In this section, we explain the details of the depth-based CNN. Other than 3D-mask PA, all known PAs, such as printed paper and display, have an obviously different depth compared to the live faces. Therefore, developing a robust depth estimator can benefit the face PAD.

Based on [17], we believe that high-frequency information of face images is crucial for face PAD, and resizing images may lead to a loss of high-frequency information. Therefore, to be able to handle face images with different sizes, we propose to maintain the original image size in training the CNN for depth estimation. That is, we train a fully convolutional network (FCN) whose parameters are independent to the size of input face images. The input is face images, and the output is the corresponding depth maps. For the live faces, the depth information is from the 3D face shapes estimated using a state-of-the-art 3D face model fitting algorithm

[24–26, 35]. For the spoof faces, the depth information is the flat plain, as assumed by the attack medium's geometry, e.g., screen, paper.

Generating the depth labels We represent the live face with the dense 3D shape $\mathbf{A}$ as $\begin{pmatrix} x_1 & x_2 & \cdots & x_Q \\ y_1 & y_2 & \cdots & y_Q \\ z_1 & z_2 & \cdots & z_Q \end{pmatrix}$ where z denotes the depth information of the face, and Q is the number of 3D vertices.

Given the face image, the 3D face model fitting algorithm [24] can estimate the shape parameters $\mathbf{p} \in \mathbb{R}^{1\times 228}$ and projection matrix $\mathbf{m} \in \mathbb{R}^{3\times 4}$. We then use 3DMM model [5] to compute the dense 3D face shape $\mathbf{A}$ by

$$\mathbf{A} = \mathbf{m} \cdot \begin{bmatrix} \bar{\mathbf{S}} + \sum_{i=1}^{228} p^i \mathbf{S}^i \\ \mathbf{1}^\mathsf{T} \end{bmatrix}, \tag{8.1}$$

where $\bar{\mathbf{S}}$ is the mean shape of the face, and $\mathbf{S}^i$ are the PCA shape bases representing identification variations, e.g., tall/short, light/heavy, and expression variations, e.g., mouth opening, smile.

After we compute the 3D dense shape of the face, the depth map composes of the z-value for Q vertices from the shape $\mathbf{A}$. In order to obtain a smoothing and consistent depth map from discrete z-values from Q vertices, the z-buffering algorithm [39] is applied, and the "texture" of the objects is imported as the depth information (i.e., z-values). To note that, input faces with different sizes would lead to a different range for z-values, mostly proportional to the face size. Hence, the depth map $\mathbf{M}$ needs to be normalized before being used as the label for CNN training. In our case, we use the max-min method for normalization.

Examples of depth maps are shown in Fig. 8.4. For spoof faces as well as the background area in the live faces, the z-value is equal to 0. Note that for some print attacks, it is possible that the papers are bent. Since it is hard to estimate the actual amount of bending, we also treat the ground truth depth of bending papers as the flat plain.

Depth map for classification The proposed FCN can estimate a depth map for a face image. Since the depth maps used to supervise the training can distinguish between live and spoof images, the estimated depth maps should also have the capability to classify live versus spoof. To leverage this capability, we train SVM classifiers using the estimated depth maps of the training data.

Specifically, to ensure that the input dimension of SVM is of the same size, the depth map $\mathbf{M}$ is overlaid with a fixed $N \times N$ grid of cells. We compute a mean depth of each local cell and generate a N^2-dim vector, which is fed to the SVM with RBF kernel. Given that resizing the depth map might lose information, we propose to train multiple SVMs with different sizes of N. To properly determine the number of SVMs, we adopt a Gaussian mixture model to fit the distribution of input image sizes. During the testing stage, we feed the testing sample to the SVM, whose input size N is closest to the sample.

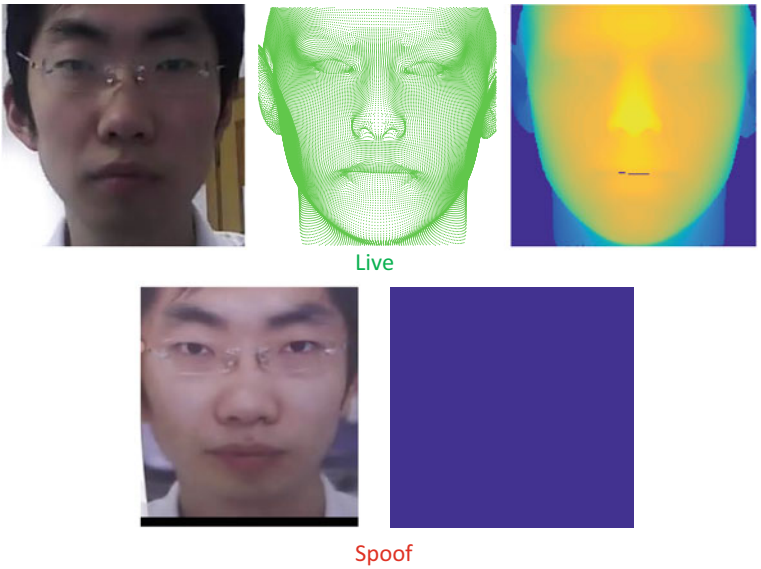

Fig. 8.4 Depth labels for depth-based CNN learning. A live face image, a fitted face model, and the depth label (top row). A spoof face image and the flat plain depth (bottom row)

Moreover, we can leverage the temporal information given a face video input. For live videos, the depth changes little over time, while the depth of spoof ones can change substantially due to noisy estimation and involuntary hand movement while holding spoof mediums. Hence for a video, we first compute a N^2-dim vector for each frame, and then compute standard deviation of the estimated depth maps of the video. The final feature of a frame feeding to SVM is a $2N^2$-dim vector. Given the SVM output of all frames, we use their average as the final score of the video.

8.3.1.3 CNN Architecture

A detailed network structure of the patch- and depth-based CNN v1 is illustrated in Table 8.1.

Patch-based CNN A total of five convolutional layers are used followed by three fully connected layers. Following every convolutional layer, we use a batch normalization, ReLU, and pooling layers. Softmax loss is utilized in CNN training. Given a training image, we initially detect the face and then crop the face region based on eye positions. After that, several patches are extracted randomly from the face image, such that all patches have the same fixed size. We avoid any rescaling to the original face images for the purpose of maintaining the spoof patterns within

Table 8.1 **a** Network structure of patch-based CNN and depth-based CNN. Red texts represent the output of the CNNs. Every convolution layer is cascaded with a ReLU layer. Note that the input size for patch-based CNN is fixed to be 96×96. The input size for depth-based CNN is varied from sample to sample. For simplicity, we show the case when the input size is 128×128. **b** The network structure of patch- and depth-based CNN v2. Red texts represent the output of the CNN. Every convolution layer is cascaded with a batch normalization and ReLU layer. Note that the input face image for the CNN v2 is normalized to 256×256

(*a*)

Patch-based CNN			*Depth-based CNN*		
Layer	Filter/Stride	Output Size	Layer	Filter/Stride	Output Size
			Conv-11	$3 \times 3/1$	$128 \times 128 \times 64$
Conv-1	$5 \times 5/1$	$96 \times 96 \times 50$	Conv-12	$3 \times 3/1$	$128 \times 128 \times 64$
BN-1		$96 \times 96 \times 50$	Conv-13	$3 \times 3/1$	$128 \times 128 \times 128$
MaxPooling-1	$2 \times 2/2$	$48 \times 48 \times 50$	MaxPooling-1	$2 \times 2/2$	$64 \times 64 \times 128$
			Conv21	$3 \times 3/1$	$64 \times 64 \times 128$
Conv-2	$3 \times 3/1$	$48 \times 48 \times 100$	Conv-22	$3 \times 3/1$	$64 \times 64 \times 256$
BN-2		$48 \times 48 \times 100$	Conv-23	$3 \times 3/1$	$64 \times 64 \times 160$
MaxPooling-2	$2 \times 2/2$	$24 \times 24 \times 100$	MaxPooling-2	$2 \times 2/2$	$32 \times 32 \times 160$
Conv-3	$3 \times 3/1$	$24 \times 24 \times 150$			
BN-3		$24 \times 24 \times 150$	Conv-31	$3 \times 3/1$	$32 \times 32 \times 128$
MaxPooling-3	$3 \times 3/2$	$12 \times 12 \times 150$	ConvT-32	$6 \times 6/1$	$37 \times 37 \times 128$
Conv-4	$3 \times 3/1$	$12 \times 12 \times 200$			
BN-4		$12 \times 12 \times 200$	Conv-41	$3 \times 3/1$	$37 \times 37 \times 128$
MaxPooling-4	$2 \times 2/2$	$6 \times 6 \times 200$	ConvT-42	$6 \times 6/1$	$42 \times 42 \times 128$
Conv-5	$3 \times 3/1$	$6 \times 6 \times 250$			
BN-5		$6 \times 6 \times 250$	Conv-51	$3 \times 3/1$	$42 \times 42 \times 160$
MaxPooling-5	$2 \times 2/2$	$3 \times 3 \times 250$	ConvT-52	$6 \times 6/1$	$47 \times 47 \times 160$
FC-1	$3 \times 3/1$	$1 \times 1 \times 1000$			
BN-6		$1 \times 1 \times 1000$	Conv-61	$3 \times 3/1$	$47 \times 47 \times 320$
Dropout	0.5	$1 \times 1 \times 1000$	ConvT-62	$6 \times 6/1$	$52 \times 52 \times 320$
FC-2	$1 \times 1/1$	$1 \times 1 \times 400$			
BN-7		$1 \times 1 \times 400$			
FC-3	$1 \times 1/1$	$1 \times 1 \times 2$	Conv-71	$3 \times 3/1$	$52 \times 52 \times 1$

(*b*)

Patch and Depth-based CNN v2		
Layer	Filter/Stride	Output Size
Conv-0	$3 \times 3/1$	$256 \times 256 \times 32$
MaxPooling-0	$3 \times 3/2$	$128 \times 128 \times 32$
Conv-1	$3 \times 3/1$	$128 \times 128 \times 32$
Conv-2	$3 \times 3/1$	$128 \times 128 \times 25$
Conv-3	$3 \times 3/1$	$128 \times 128 \times 32$
MaxPooling-1	$3 \times 3/2$	$64 \times 64 \times 32$
Conv-4	$3 \times 3/1$	$64 \times 64 \times 32$
Conv-5	$3 \times 3/1$	$64 \times 64 \times 25$
Conv-6	$3 \times 3/1$	$64 \times 64 \times 32$
MaxPooling-2	$3 \times 3/2$	$32 \times 32 \times 32$
Conv-7	$3 \times 3/1$	$32 \times 32 \times 32$
Conv-8	$3 \times 3/1$	$32 \times 32 \times 25$
Conv-9	$3 \times 3/1$	$32 \times 32 \times 32$
Concat	MaxPooling-1 + MaxPooling-2 + Conv-9	
Conv-10	$3 \times 3/1$	$32 \times 32 \times 32$
Conv-11	$3 \times 3/1$	$32 \times 32 \times 25$
Conv-12	$3 \times 3/1$	$32 \times 32 \times 2$

the extracted patches. If the face image is a live face, we assign all of its patches a binary label of 1. If the face is a spoof face, the labels of patches are 0.

During testing, we extract patches in the same manner as training. The patch-based CNN will produce spoof scores for every patch in the range of 0–1. The final result of the image is the average spoof score of all patches. If the presentation attack is in the video format, we compute the average spoof score across all frames.

Depth-based CNN We employ a FCN to learn the nonlinear mapping function $f(\mathbf{I}; \Theta)$ from an input image $\mathbf{I}$ to the corresponding depth map $\mathbf{M}$, where Θ is the network parameter. Following the setting in Sect. 8.3.1.1, we use $HSV + YC_bC_r$ features as the CNN input. The depth label $\mathbf{M}$ is obtained in the approach described in the previous subsection. Our FCN network has a bottleneck structure, which contains two parts, downsampling part and upsampling part, as shown in Table 8.1. The downsampling part contains six convolution layers and two max-pooling layers; the upsampling part consists of $five$ convolution layers which sandwich $four$ transpose convolution layers for the upsampling purpose. This architecture composes of only

convolution layers without fully connected layer, and each layer is followed by the ReLU layer. We define the loss function as the pixel-level Euclidean loss,

$$\arg\min_{\Theta} J = \|f(\mathbf{I};\Theta) - \mathbf{M}\|_F^2 . \tag{8.2}$$

8.3.2 *Patch- and Depth-Based CNN v2*

As we evaluate the patch- and depth-based CNN v1, we notice several drawbacks of this model. First, it is not very time-efficient. With N random patches and one whole image to go through the CNNs for each sample, the system requires approximately $1 + \frac{N}{2}$ seconds to process each frame, which is not suitable for mobile applications such as phone unlocking. To reduce the time cost, we revisit the patch-based approach and propose a fully convolution network that learns the same patch-based features. Secondly, CNN v1 is trained with limited amount of data, where most of them are captured in a constrained environment. It could make wrong estimations for input samples with extreme illuminations, poses, and other unknown factors. To handle this real-world issue, we deploy a simple but effective strategy of continuous updating that can improve the generalization of the system by a large margin. The overall architecture of the CNN v2 is shown in Fig. 8.5.

8.3.2.1 Revisit the Patch-Based CNN

We mention several motivations to use patches instead of the full face in CNN in Sect. 8.3.1.1, and one of the major reasons is to prevent overfitting of the CNN training. However, during the testing time, we need to sample sufficient amount of random patches in order to maintain a consistent performance, which can be very time-consuming. We revisit the design of the patch-based CNN. For the framework of CNN, the effective receptive field for a certain level of convolution layer is constrained. For example, the conv-13 in Tabe 8.1 has a receptive field of 5×5 patch in

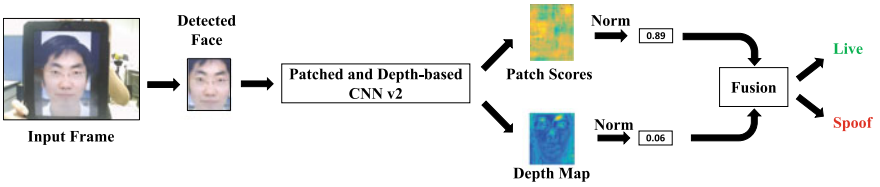

Fig. 8.5 Architecture of the advanced face PAD approach

the original input. With a specified depth, the output of a fully convolutional network (FCN) is locally receptive and hence essentially *patch-based*.

Therefore, we redesign the patch-based CNN as a fully convolutional network, where the input of the network is the cropped face, and the output of the network is a binary mask. Each element of the mask indicates the spoofness of a local patch whose size is the same as that of the receptive field. For a spoof face, by assuming the whole region of the face as well as its close surrounding are all from the spoof medium, such as printed paper and digital screen, the binary mask is an *one* map. And for a live face, the binary mask is a *zero* map.

Despite the patch-based CNN shares similar architecture as the depth-based CNN, they still learn distinct features for detecting PA. The patch-based CNN focuses on generic spoofing features that exist on all local regions of the image, while the depth-based CNN focuses on face-dependent features that only exist within face regions, such as eye corners, cheek, and jaw lines.

8.3.2.2 Maps for Classification

Patch-based CNN provides a binary mask map to indicate the spoofness of each local patch, and depth-based CNN provides an estimated depth map for the whole face region. Since the given labeling (i.e., binary mask or depth map) itself is discriminative with respect to live versus spoof, it might not be necessary to use an additional classifier (e.g., SVM) for classification. To convert these two maps into a score for decision making, we simply compute the L_2 norm of each map and sum them up with an assigned weight, as shown in Eq. 8.3.

$$score = \alpha \|\mathbf{M}\|_2^2 + \|\mathbf{D}\|_2^2 . \tag{8.3}$$

8.3.2.3 Model Compression

To utilize the trained face PAD CNN model for authentication in mobile and embedded devices, we should make the CNN model compatible with the computational power on those devices. There are many research papers for compressing the CNN models and reducing their computational cost. The model compressing methods [13] can be categorized into four main groups: (1) parameter pruning and sharing [51], (2) knowledge distillation [21], (3) transferred convolutional filters [48], (4) low-rank factorization [22].

In this work, our objective is to find the model with the minimum computational requirement; hence, we design the model compression as a search algorithm. We utilize a new greedy method similar to the binary search for finding the minimum number of filters needed in each layer. We make a development set for evaluating the performance of the compressed models with our greedy method. To find the minimum size of the network with acceptable performance on the development set, we iteratively reduce the number of filters by half while keeping the number of layers

fixed and retraining the network. We stop this process when the CNN model cannot achieve acceptable performance on the development set, as an indication of low capacity of the CNN model for the face anti-spoofing task. By applying this method, we reduce the size of the CNN model by 160 times from $\sim$ 80 Mb to 0.5 Mb while achieving similar performance.

8.3.2.4 CNN v2 Architecture

A detailed network structure of the patch- and depth-based CNN v2 is illustrated in Table 8.1. With the same input to the network, the new patch-based CNN and depth-based CNN can share the weights with each other, since they are trained for the same purpose. We combine these two networks into one to further reduce the computation time by half. After model compression, a total of nine convolution layers with three max-pooling layers are used to extract spoofing features at different levels of scales. Then, we adopt a shortcut connection to concatenate the feature maps of each pooling layer with the size normalized as 32. The concatenated features are sent to three additional convolution layers. The final output of the network is two 32×32 maps, where the first map is supervised by the zero/one map for learning the patch-based features, and the second map is supervised by the depth map for learning the depth-based face-dependent features. The maximum receptive field of the CNN v2 is 72. The L_1 loss is utilized in CNN training. Following every convolution layer, we use a batch normalization layer and ReLU layer.

Given a testing image, we initially detect the face and then crop the face region based on eye positions. The cropped face is utilized in the CNN v2 to produce the binary mask map $\mathbf{M}$ as well as the depth map $\mathbf{D}$. The final score for the testing sample is a weighted average of the map norms, as shown in Eq. 8.3.

8.3.2.5 CNN System Updating

To improve performance and increase the robustness of the network, we utilize iterative update training by incorporating failure cases from the previous model. This process is shown in Fig. 8.6. We begin with a trained model and its corresponding set of training data. The performance of the trained model is qualitatively analyzed by collecting failure cases using our PC and Android demo applications. These failure cases are then analyzed, specifically considering if there are patterns that are common to the experienced failures such as low illumination, reflective glare on spoofs, washing out due to excessive illumination, and extreme pose, among others. The collected failure case images are then added into the training data, and the model is updated using a random shuffle of the previous and newly collected data. In this way,

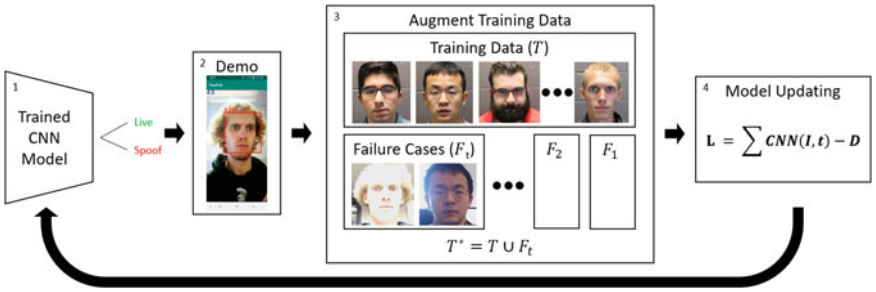

Fig. 8.6 Iterative updating of the trained model using the most recent failure cases collected by the PC and Android apps allows for targeted improvements for situations in which the model fails. The updating process begins with a model trained on our training dataset. Newly collected data is added to the current training data. This significantly and quickly improves the trained model without unnecessary effort to collect unimportant data

we enforce that the model performs similarly on previous success cases and previous failure cases, while improving performance on the most recent failure cases. As we repeat the updating process multiple times, it becomes more difficult to collect failure cases, indicating that the model has become more robust to its previous weaknesses.

8.4 Experiments

8.4.1 Database

We evaluate our proposed method on two PAs: print and replay attacks, using five benchmark databases: CASIA-MFSD [60], MSU-USSA [41], replay attack [16], OULU-NPU [11], and SiW [34].

CASIA-MFSD: This database contains 50 subjects and 12 videos for each subject under *three* different image resolutions and varied lightings. Each subject includes *three* different spoof attacks: replay, warp print, and cut print attacks. Due to the diversity of the spoof types, many previous works [40, 52] that leverage the motion cues such as eye blinking or shape deformation would fail on this dataset. This dataset partitions the subject space and uses 20 subjects for training and 30 subjects for testing.

MSU-USSA: As one of the largest public face spoofing databases, MSU-USSA contains 1000 in the wild live subject images from the weakly labeled face Database and creates *eight* types of spoof attacks from different devices such as smart phones, personal computers, tablets, and printed papers. This dataset covers images under different illuminations, image qualities, and subject diversity.

Replay attack: This database contains 1, 300 live and spoof videos from 50 subject. These videos are divided into training, development, and testing sets with 15, 15, and 20 subjects, respectively. The videos contain two illumination conditions: controlled and adverse. Given the print and replay attacks in this set, the database also divides the attacks into two more types based on whether they use a support to hold the spoof medium, or if the attack is held by a person.

OULU-NPU: This more recent database is comprised of 4920 live and spoof videos captured of 55 subjects using six mobile phone cameras in $three$ sessions with varying illumination conditions and scene backgrounds. Unlike earlier databases, this uses $1080p$ videos to accommodate higher quality images' increasing prevalence in society. Four testing protocols are defined to evaluate a network's performance under differing situations such as generalization in leave-one-out testing.

8.4.2 Experimental Parameters and Setup

In CNN v1, we use Caffe toolbox [23] to implement the patch-based CNN. The learning rate is set as 0.001, decay rate as 0.0001, momentum as 0.99, and batch size as 100. Before being fed into the CNN, the face samples are normalized by subtracting the mean face of training data. Since CASIA and replay attack are video datasets, we only extract two random patches per frame for training. For the images in MSU-USSA, we extract 64 patches from each live face region and $eight$ patches from each spoof face region. For CASIA and MSU-USSA, a fixed patch size of 96×96 is used. For replay attack, given its low image resolution, the patch size is 24×24. To accommodate the difference in patch sizes, we remove the first two pooling layers for the patch-based CNN. For the depth-based CNN, we use TensorFlow [1], with the learning rate of 0.01 and batch size of 32. The patches are also normalized by subtracting the mean face of training data. When generating the depth labels, we normalize the depth in the range of 0–1. We use the weighted average of two streams' scores as the final score of our proposed method, where the weights are experimentally determined.

In CNN v2, we use TensorFlow to implement all parts, with the learning rate of 0.01 and batch size of 32. The input face samples are normalized to be $256 \times 256 \times 3$. The depth maps are also normalized to the range of 0–1 with the size of 32. We use the weighted average of two feature maps as the final score as mentioned in Sect. 8.3.2.2. Based on the experiments, the final α and β are set to be 0.5 and -1.2, respectively, for all of the following experiments.

Our experiments follow the protocol associated with each of the five databases. For each database, we use the training set to learn the CNN models and the testing set for evaluation in terms of equal error rate (EER) and half total error rate (HTER).

8.4.3 Ablation Study

8.4.3.1 Patch-Based CNN Analysis

In Sect. 8.3.1.1, we explore several input feature maps to train the patch-based CNN v1, which include different combinations of color spaces, a pixel-wise *LBP* map, and high-frequency patches. For all of the experiments, we first detect and then crop the face for a given frame. After that we convert the face image into a new feature map as seen in Fig. 8.3, which will then be used to extract patches. Table 8.2 presents the results on CASIA-FASD when using different combinations of input feature maps. Based on our experiments, we only show the best four combinations of features in this table. From these results, we can clearly see that the $HSV + YC_bC_r$ features have a significant improvement in performance compared to the other features with an EER of 4.44% and an HTER of 3.78%. Moreover, adding an *LBP* map to the $HSV + YC_bC_r$ has a negative impact to the CNN learning, which reduces the performance of using $HSV + YC_bC_r$ only by 2.31% HTER. Similarly, when training the patch-based CNN with high-frequency data in the $HSV + YC_bC_r$ images, it also reduces the performance by 1.79% HTER. This shows that the low frequencies may also provide discriminative information to anti-spoofing.

8.4.3.2 Depth-Based CNN Analysis

The depth map results of CNN v1 on the CASIA-FASD testing set are shown in Fig. 8.7. The CNN is attempting to predict the face-like depth of a live face, i.e., higher values in the depth map, while the predicted depth of the spoof images to be flat, i.e., lower values in the depth map. Due to the difficulty of the problem, the estimated depth is not perfect, compared to the depth label shown in Fig. 8.4. However, we can still find a clear distinction between the depth maps of the live images and those of the spoof images. In the spoof image, there might be certain areas that suffer more degradation and noise from the spoof attack. As we can see from Fig. 8.7, our CNN is still trying to predict some areas with high values in the depth map. However, overall depth patterns of spoof samples are far from those of live samples so that the SVM can learn their difference. Hence, training a CNN for depth estimation is beneficial to

Table 8.2 EER (%) and HTER (%) of CASIA-FASD, when feeding different features to patch-based CNN

Feature	EER (%)	HTER (%)
YC_bC_r	4.82	3.95
$YC_bC_r + HSV$	**4.44**	**3.78**
$YC_bC_r + HSV + LBP$	7.72	6.09
$(YC_bC_r + HSV)_H$	9.58	5.57

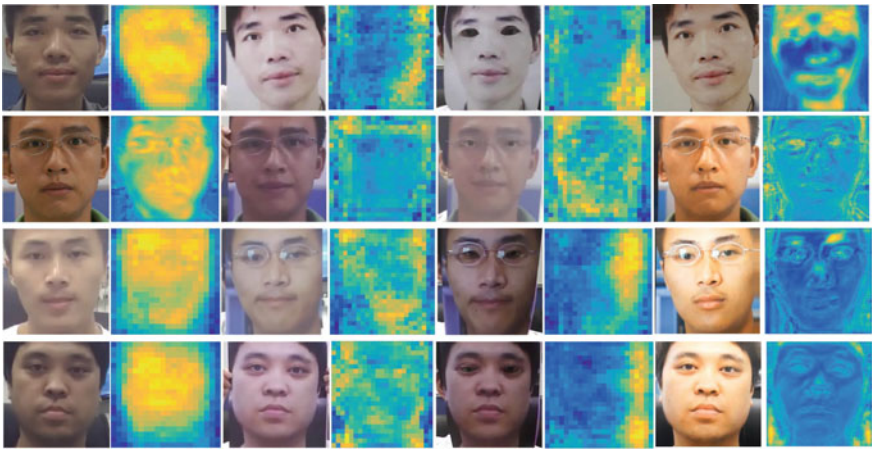

Fig. 8.7 Depth estimation on CASIA-FASD testing subjects. The first two columns are the live images and their corresponding depth maps, and the rest six columns are three different types of spoof attacks (print, cut print, and video attacks) and their corresponding depth maps

face anti-spoofing. Fig. 8.8 shows the mean and standard deviation of the estimated depth maps for all live faces and spoof faces in replay attack. The differences of live vs. spoof in both mean and standard deviation demonstrate the discriminative ability of depth maps, as well as support the motivation of feature extraction for SVM in Sect. 8.3.2.2.

8.4.3.3 Fusion Analysis

We extensively analyze the performance of our patch-based and depth-based CNN v1 on CASIA-FASD and report frame-based performance curves as seen in Fig. 8.9. As mentioned earlier, CASIA-FASD has three different video qualities and three different presentation attacks, which we use to highlight the differences of our proposed CNN streams. For the low-quality images, the patch-based method achieves an EER of 2.78%. For the same quality, we notice that depth-based CNN performs better, which is understandable since the relative depth variation of frontal-view face image is very small compared to the far distance when a low-quality face image is captured. For the normal quality, the fusion of both methods has a large positive impact on the final result, which can be seen from the ROC curves. The result of both methods on high-quality videos is reasonably good, and therefore, fusion will maintain the same performance. It is clear that the depth-based method struggles when the face images are lower in resolution, and vice-versa for the patch-based method. On the other hand, the patch-based method suffers with high resolution, and vice-versa for the depth-based method. Therefore, the fusion of both methods will strengthen the weak part of either one.

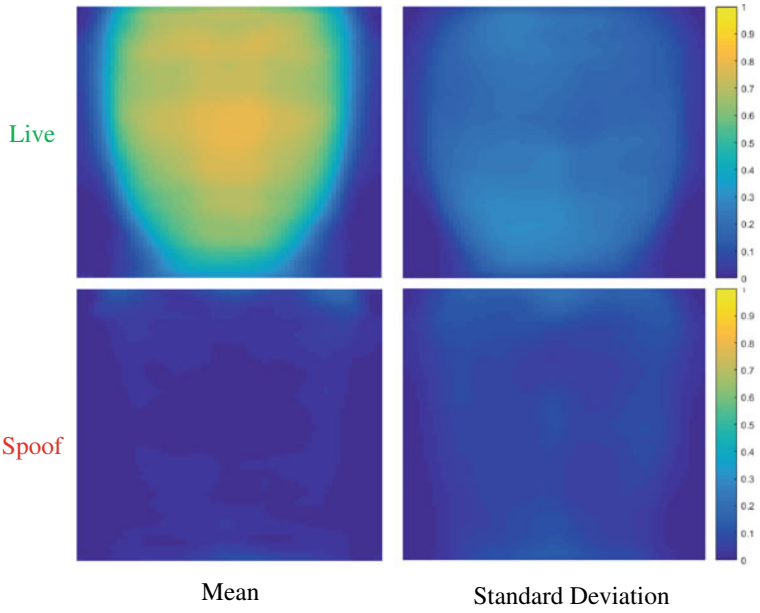

Fig. 8.8 Mean and standard deviation of the estimated depth maps of live and spoof faces, for all testing samples in replay attack. Note the clear differences in both the mean and standard deviation between the two classes

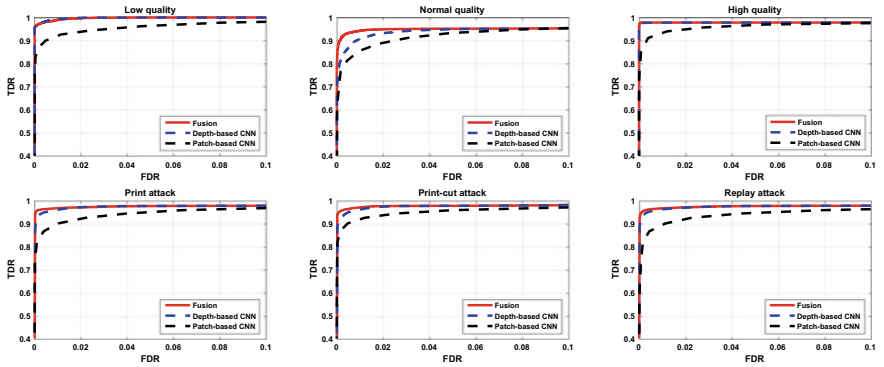

Fig. 8.9 Frame-based ROC curves on CASIA-FASD comparing the fusion method with the patch-based and depth-based CNNs

When analyzing the three different presentation attacks in CASIA-FASD with our proposed methods, the most successfully detected attack is the video replay attack. It is worthy to note that, since the ROC curve of every attack is an average of the three different video qualities, the difference among the three attacks is not large. For the fusion results, the best gain can be seen in the print attacks compared to the results of the two methods independently.

8.4.3.4 Patch- and Depth-Based CNN v2 Analysis

Similarly, we evaluate the performance of the patch- and depth-based CNN v2. In CNN v2, we utilize the CNN to estimate two maps, a binary mask of spoofness, and a depth map. We train a CNN with binary mask supervision only and a CNN with depth map supervision only to validate the effectiveness of fusing the two feature maps. The test is conveyed on CASIA-FASD database. Without the supervision of depth map, the CNN obtains 6.7% as the EER and 6.3% as the HTER; without the supervision of binary mask, the CNN obtains 25.6% as the EER and 20.4% as the HTER. By combining the two streams, the CNN v2 can achieve the best performance of 4.4% as the EER and 4.6% as the HTER.

To further show the effectiveness of the continuous updating strategy, we collect a private testing set. To continuously update the model, we use the face PAD demo system with CNN v2 to capture failure cases from $five$ subjects, none of which are included in the private testing set. The model without updating obtains 31.2% as the EER and 31.1% as the HTER, while the model with updating achieves 6.2% as the EER and 5.4% as the HTER, which demonstrates a large margin of improvement.

8.4.4 Experimental Comparison

We compare the proposed method with the state-of-the-art CNN-based methods on CASIA-FASD. Table 8.3 shows the EER and HTER of six face anti-spoof methods. Among different methods in Table 8.3, the temporal features are utilized in a long short-term memory (LSTM) CNN [57], the holistic features are extracted for classification in [59], CNN is used for the feature extraction in [33], and after applying PCA to the response of the last layer, SVM is utilized for classification. According to Table 8.3, our method outperforms others in both EER and HTER. This shows the combination of local and holistic features contain more discriminative information. Note that even though depth-based CNN alone has larger errors, its fusion with patch-based CNN still improves the overall performance. For CNN v2, it shows a perfect performance on replay database and the high-resolution part of CASIA dataset with EER and HTER to be 0. However, it performs worse on the low-resolution part of the CASIA dataset, and thus the overall EER and HTER are slightly worse than CNN v1. Because of the superior performance on the first two part and its time efficiency, we still regard CNN v2 as a better model and use CNN v2 to conduct further experiments.

Table 8.3 EER (%) and HTER (%) on CASIA-FASD

Method	EER (%)	HTER (%)
Fine-tuned VGG-face [33]	5.20	–
DPCNN [33]	4.50	–
[59]	4.92	–
CNN [57]	6.20	7.34
[10]	6.2	–
[49]	3.14	–
[8]	2.8	–
[57]	5.17	5.93
Haralick features [2]	–	1.1
Moire pattern [43]	–	0
Patch-based CNN	4.44	3.78
depth-based CNN	2.85	2.52
Patch- and depth-based CNN v1	**2.67**	**2.27**
Patch- and depth-based CNN v2	4.4	4.6

Table 8.4 EER (%) and HTER (%) on MSU-USSA

Method	EER (%)	HTER (%)
[41]	3.84	–
Patch-based CNN	0.55 ± 0.26	0.41 ± 0.32
depth-based CNN	2.62 ± 0.73	2.22 ± 0.66
Patch and depth-based CNN v1	0.35 ± 0.19	0.21 ± 0.21
Patch and depth-based CNN v2	$\mathbf{0} \pm 0$	$\mathbf{0} \pm 0$

We also test our method on the MSU-USSA database. Not many papers report results in this database because it is relatively new. Table 8.4 compares our results with [41] which analyzes the distortions in spoof images and provides a concatenated representation of LBP and color moment. In comparison with [41], our patch-based CNN already achieves 89% reduction of EER. The complementariness of depth-based CNN further reduce both the EER and HTER.

On the replay attack database [16], we compare the proposed method with three prior methods in Table 8.5. For the CNN v1, although our EER is similar to the prior methods, the HTER of our method is much smaller, which means we have fewer false acceptance and rejection. Moreover, though the fusion does not significantly reduce the EER and HTER over the depth-based CNN, we do observe an improvement in the AUC from 0.989 in patch-based CNN to 0.997 in the fusion. Additionally, our CNN v2 is able to achieve perfect performance on the replay attack dataset, demonstrating the enhanced capability of the CNN v2.

Table 8.5 EER (%) and HTER (%) on replay attack

Method	EER (%)	HTER (%)
Fine-tuned VGG-face [33]	8.40	4.30
DPCNN [33]	2.90	6.10
[59]	2.14	–
[10]	0.4	2.9
[8]	0.1	2.2
Moire pattern [43]	–	3.3
Patch-based CNN	2.50	1.25
Depth-based CNN	0.86	0.75
Patch- and Depth-based CNN v1	0.79	0.72
Patch- and Depth-based CNN v2	0	0

Table 8.6 Performance of the proposed method and SOTA face anti-spoofing methods during cross-dataset evaluation on current face anti-spoofing datasets. The HTER is reported. Results for other works are reported for cross-dataset testing between CASIA-FASD and replay attack. Results for OULU and the private test set are the HTER over the entire test set. The results for our models are only reported if the model evaluated was not trained on the corresponding training data for a given test set

Algorithm	CASIA-FASD	replay attack	OULU	Private test
Motion [19]	47.9	50.2	–	–
Spectral cubes [45]	50.0	34.4	–	–
Color LBP [10]	35.4	37.9	–	–
Color texture [7]	37.7	30.3	–	–
Color SURF [9]	**23.2**	26.9	–	–
Boulkenafet [9]	39.2	**9.6**	–	–
Liu [34]	27.6	28.4	–	–
CNN v2 (CASIA baseline)	–	42.0	25.3	26.7
CNN v2 (Replay baseline)	43.2	–	36.2	27.8
CNN v2 (Without updating)	36.1	34.7	33.1	31.1
CNN v2 (With updating)	**23.2**	**15.4**	**0.0**	**5.4**

Table 8.6 shows the performance of the CNN v2 compared to SOTA performance for cross-dataset evaluation. In this cross-dataset evaluation scenario, the HTER for all methods is significantly poorer than in the intra-dataset evaluation scenario, as is evident in our best performance in HTER of 23.2% compared to 2.3% for CASIA-FASD and 15.4% compared to 0.0% for replay attack. Our CASIA baseline and replay baseline performance are for the CNN v2 trained on CASIA-FASD and replay attack, respectively. When trained only on this low-resolution data, the cross-dataset performance is poor. The CNN v2 without updating, which was trained on a

larger dataset of mostly $1080p$ images, lags slightly behind the SOTA performance. However, when we incorporate the iterative updating of the network, we are able to achieve SOTA performance for cross-dataset evaluation, except in the case of the replay attack dataset. However, the SOTA work that performs best on replay attack performs much worse than our CNN v2 with updates on CASIA-FASD, indicating that our CNN v2 with updating is much more stable and able to generalize well. This further demonstrates the value of iterative updates using failure cases to improve the robustness of the network, even in the case of cross-dataset evaluation.

8.4.5 Model Size and Running Time

Figure 8.10 shows the Android demo app under three different situations. The Android demo has two major functions, testing the performance of the trained model and collecting failure cases for the current model. This is accomplished via three modes in the demo: (*i*) normal mode, (*ii*) live failure mode, and (*iii*) spoof failure mode. A prediction of live will draw a green bounding box around the face. In normal mode, the score is displayed to the screen. In live failure mode, it is assumed that any detected faces are live. A prediction of spoof will save the image to the phone's

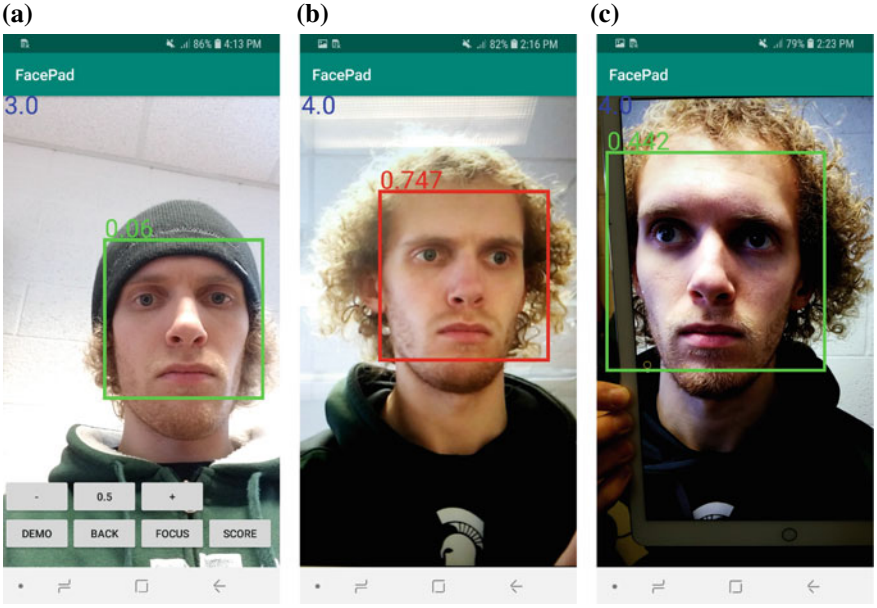

Fig. 8.10 Screenshot of the Android mobile phone demo. A score of 0 indicates live, and a score of 1 indicates spoof. Shown are **a** correct detection of a live face, **b** correct detection of a spoof attack, **c** failed detection of a spoof attack and subsequent capture of the failure case

storage as a live failure case. In spoof failure mode, it is assumed that any detected faces are spoof. A prediction of live will save the image to the phone's storage as a spoof failure case.

The demo mode and failure case threshold can be changed via the in-app settings (shown in Fig. 8.10). Modifying the failure case threshold, let us tune the difficulty for a failure case to occur by requiring higher confidence as the updating process matures. These settings also allow for hiding the raw score value, switching between the front and rear cameras of the device and refocusing the camera. Similar settings are available on the PC demo for collecting failure cases.

In an iterative method, the failure cases collected by both the PC and Android demos are added to the training set, and the model is updated with these additional images. This allows for rapid improvement of common failure cases across any of our demo-enabled devices. Often a failure case from a previous iteration will be correctly classified by the updated model, thereby reducing the number of failure cases collected each iteration. The reduced number of failure cases indicates that the updated model is becoming increasingly robust against attacks it was previously weak to. As shown in Table 8.6, the model with updating performs significantly better than the non-updated model.

Due to the limited computation ability of smartphone devices compared to PCs, we must reduce the memory and processing time of the trained model for the Android demo. To do this, we reduce the number of kernels in the convolutional layers until an appropriately small and fast, but still accurate model is produced. This improves the responsiveness of the Android app by doubling its FPS, but requires a small degradation in performance. Finally, we are able to achieve *nine* FPS on the PC application without using the GPU, which increases to 30 FPS when we enable the GPU. We are unable to utilize the GPU on Android smartphones, and hence are limited to *four* FPS for the reduced CNN v2 model or 1–2 FPS for the full CNN v2 model.

8.5 Conclusions

In this chapter, we introduce a novel solution for mobile face PAD system via fusing patch-based CNN and depth-based CNN. Unlike the prior PAD methods that use the full face and single labels to train and detect presentation attacks, we leverage both supervisions for local patches from the same face and estimation of the face depth to distinguish the spoof from live faces. The first CNN stream is based on patch appearance extracted from face regions. This stream demonstrates its robustness across all presentation attacks, especially on lower-resolution face images. The second CNN stream is based on face depth estimation using the full face image. We prove it can improve the performance via fusing two CNNs. We further improve its mobile performance via combining two CNNs into one, optimizing the network structures and applying the strategy of continuous learning. The experiments show that the proposed CNN is robust, generalized, and computationally efficient in several testing scenarios, either intra-testing on the commonly used database, or testing sample from real-world hard cases.

References

1. Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M, Kudlur M (2016) Tensorflow: a system for large-scale machine learning. In: OSDI, vol 16, pp 265–283
2. Agarwal A, Singh R, Vatsa M (2016) Face anti-spoofing using Haralick features. In: 2016 IEEE 8th international conference on biometrics theory, applications and systems (BTAS). IEEE, pp 1–6
3. Atoum Y, Liu Y, Jourabloo A, Liu X (2017) Face anti-spoofing using patch and depth-based CNNs. In: 2017 IEEE international joint conference on biometrics (IJCB). IEEE, pp 319–328
4. Bao W, Li, H, Li, N, Jiang W (2009) A liveness detection method for face recognition based on optical flow field. In: International conference on image analysis and signal processing, 2009. IASP 2009. IEEE, pp 233–236
5. Blanz V, Vetter T (2003) Face recognition based on fitting a 3D morphable model. IEEE Trans Pattern Anal Mach Intell 25(9):1063–1074
6. Blunsom P, Grefenstette E, Kalchbrenner N (2014). A convolutional neural network for modelling sentences. In Proceedings of the 52nd annual meeting of the association for computational linguistics
7. Boulkenafet Z, Komulainen J, Hadid A (2016) Face spoofing detection using colour texture analysis. IEEE Trans Inf Forensics Secur 11(8):1818–1830
8. Boulkenafet Z, Komulainen J, Hadid A (2017) Face antispoofing using speeded-up robust features and fisher vector encoding. IEEE Signal Process Lett 24(2):141–145
9. Boulkenafet Z, Komulainen J, Hadid A (2018) On the generalization of color texture-based face anti-spoofing. Image Vis Comput 77:1–9
10. Boulkenafet Z, Komulainen J, Hadid A (2015) Face anti-spoofing based on color texture analysis. In: 2015 IEEE international conference on image processing (ICIP). IEEE, pp 2636–2640
11. Boulkenafet Z, Komulainen J, Li L, Feng X, Hadid A (2017) OULU-NPU: a mobile face presentation attack database with real-world variations. In: 2017 12th IEEE international conference on automatic face & gesture recognition (FG 2017). IEEE, pp 612–618
12. Chen W, Fu Z, Yang D, Deng J (2016) Single-image depth perception in the wild. In: Advances in neural information processing systems, pp 730–738
13. Cheng Y, Wang D, Zhou P, Zhang T (2017) A survey of model compression and acceleration for deep neural networks. arXiv:1710.09282
14. Chetty G (2010) Biometric liveness checking using multimodal fuzzy fusion. In 2010 IEEE international conference on fuzzy systems (FUZZ). IEEE, pp 1–8
15. Chetty G, Wagner M (2006) Audio-visual multimodal fusion for biometric person authentication and liveness verification. In: Proceedings of the 2005 NICTA-HCSNet multimodal user interaction workshop, vol 57. Australian Computer Society Inc., pp 17–24
16. Chingovska I, Anjos A, Marcel S (2012) On the effectiveness of local binary patterns in face anti-spoofing. In: Proceedings of the 11th international conference of the biometrics special interest group (No. EPFL-CONF-192369)
17. da Silva Pinto A, Pedrini H, Schwartz W, Rocha A (2012) Video-based face spoofing detection through visual rhythm analysis. In: 2012 25th SIBGRAPI conference on graphics, patterns and images (SIBGRAPI). IEEE, pp 221–228
18. de Freitas Pereira T, Anjos A, De Martino JM, Marcel S (2012) LBP-TOP based countermeasure against face spoofing attacks. In: Asian conference on computer vision. Springer, Berlin, Heidelberg, pp 121–132
19. de Freitas Pereira T, Anjos A, De Martino JM, Marcel S (2013) Can face anti-spoofing countermeasures work in a real world scenario? In: 2013 International conference on biometrics (ICB). IEEE, pp 1–8
20. Feng L, Po LM, Li Y, Xu X, Yuan F, Cheung TCH, Cheung KW (2016) Integration of image quality and motion cues for face anti-spoofing: a neural network approach. J Vis Commun Image Represent 38:451–460

21. Hinton G, Vinyals O, Dean J (2015) Distilling the knowledge in a neural network. arXiv:1503.02531
22. Jaderberg M, Vedaldi A, Zisserman A (2014) Speeding up convolutional neural networks with low rank expansions. arXiv:1405.3866
23. Jia Y, Shelhamer E, Donahue J, Karayev S, Long J, Girshick R, Guadarrama S, Darrell T. (2014) Caffe: convolutional architecture for fast feature embedding. In: Proceedings of the 22nd ACM international conference on Multimedia. ACM, pp 675–678
24. Jourabloo A, Liu X (2017) Pose-invariant face alignment via CNN-based dense 3D model fitting. Int J Comput Vis 124(2):187–203
25. Jourabloo A, Liu X (2015) Pose-invariant 3D face alignment. In: Proceedings of the IEEE international conference on computer vision, pp 3694–3702
26. Jourabloo A, Liu X (2016) Large-pose face alignment via CNN-based dense 3D model fitting. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4188–4196
27. Jourabloo A, Liu Y, Liu X (2018) Face de-spoofing: anti-spoofing via noise modeling. In: European conference on computer vision. Springer, Cham, pp 297–315
28. Karsch K, Liu C, Kang SB (2014) Depth transfer: depth extraction from video using non-parametric sampling. IEEE Trans Pattern Anal Mach Intell 36(11):2144–2158
29. Kollreider K, Fronthaler H, Faraj MI, Bigun J (2007) Real-time face detection and motion analysis with application in "liveness" assessment. IEEE Trans Inf Forensics Secur 2(3):548–558
30. Komulainen J, Hadid A, Pietikainen M (2013) Context based face anti-spoofing. In: 2013 IEEE sixth international conference on biometrics: theory, applications and systems (BTAS). IEEE, pp 1–8
31. Krizhevsky A, Sutskever I, Hinton GE (2012). Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105
32. Lawrence S, Giles CL, Tsoi AC, Back AD (1997) Face recognition: a convolutional neural-network approach. IEEE Trans Neural Netw 8(1):98–113
33. Li L, Feng X, Boulkenafet Z, Xia Z, Li M, Hadid A (2016). An original face anti-spoofing approach using partial convolutional neural network. In: 2016 6th international conference on Image processing theory tools and applications (IPTA). IEEE, pp 1–6
34. Liu Y, Jourabloo A, Liu X (2018) Learning deep models for face anti-spoofing: Binary or auxiliary supervision. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 389–398
35. Liu Y, Jourabloo A, Ren W, Liu X (2017) Dense face alignment. In: Proceedings of IEEE international conference on computer vision workshops, pp 1619–1628
36. Liu F, Zeng D, Zhao Q, Liu X (2018) Disentangling features in 3D face shapes for joint face reconstruction and recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 5216–5225
37. Li J, Wang Y, Tan T, Jain AK (2004) Live face detection based on the analysis of Fourier spectra. In: Biometric technology for human identification, vol 5404. International Society for Optics and Photonics, pp 296–304
38. Määttä J, Hadid A, Pietikäinen M (2011) Face spoofing detection from single images using micro-texture analysis. In: 2011 International joint conference on biometrics (IJCB). IEEE, pp 1–7
39. Matsumoto T (1991) U.S. Patent No. 5,043,922. U.S. Patent and Trademark Office, Washington, DC
40. Pan G, Sun L, Wu Z, Lao S (2007) Eyeblink-based anti-spoofing in face recognition from a generic webcamera
41. Patel K, Han H, Jain AK (2016) Secure face unlock: spoof detection on smartphones. IEEE Trans Inf Forensics Secur 11(10):2268–2283
42. Patel K, Han H, Jain AK (2016) Cross-database face antispoofing with robust feature representation. In: Chinese conference on biometric recognition. Springer, Cham, pp 611–619

43. Patel K, Han H, Jain AK, Ott G (2015). Live face video versus spoof face video: use of moiré patterns to detect replay video attacks. In: 2015 International conference on biometrics (ICB). IEEE, pp 98–105
44. Peixoto B, Michelassi C, Rocha A (2011) Face liveness detection under bad illumination conditions. In: 2011 18th IEEE international conference on image processing (ICIP). IEEE, pp 3557–3560
45. Pinto A, Pedrini H, Schwartz WR, Rocha A (2015) Face spoofing detection through visual codebooks of spectral temporal cubes. IEEE Trans Image Process 24(12):4726–4740
46. Roth J, Tong Y, Liu X (2017) Adaptive 3D face reconstruction from unconstrained photo collections. IEEE Trans Pattern Anal Mach Intell 39(11):2127–2141
47. Roth J, Tong Y, Liu X (2015) Unconstrained 3D face reconstruction. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2606–2615
48. Shang W, Sohn K, Almeida D, Lee H (2016) Understanding and improving convolutional neural networks via concatenated rectified linear units. In: International conference on machine learning, pp 2217–2225
49. Siddiqui TA, Bharadwaj S, Dhamecha TI, Agarwal A, Vatsa M, Singh R, Ratha N (2016) Face anti-spoofing with multifeature videolet aggregation. In: 2016 23rd international conference on pattern recognition (ICPR). IEEE, pp 1035–1040
50. Silberman N, Hoiem D, Kohli P, Fergus R (2012) Indoor segmentation and support inference from RGBD images. In: European conference on computer vision. Springer, Berlin, Heidelberg, pp 746–760
51. Srinivas S, Babu RV (2015) Data-free parameter pruning for deep neural networks. arXiv:1507.06149
52. Sun L, Pan G, Wu Z, Lao S (2007) Blinking-based live face detection using conditional random fields. In: International conference on biometrics. Springer, Berlin, Heidelberg pp 252–260
53. Tan X, Li Y, Liu J, Jiang L (2010) Face liveness detection from a single image with sparse low rank bilinear discriminative model. In: European conference on computer vision. Springer, Berlin, Heidelberg pp 504–517
54. Tran L, Liu X (2018) Nonlinear 3D Face Morphable Model. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 7346–7355
55. Tran L, Yin X, Liu X (2017) Disentangled representation learning GAN for pose-invariant face recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1283–1292
56. Wang D, Hoi S, Zhu J (2014) Wlfdb: Weakly labeled face databases, vol 5. Technical report
57. Xu Z, Li S, Deng W (2015). Learning temporal features using LSTM-CNN architecture for face anti-spoofing. In: 2015 3rd IAPR Asian conference on pattern recognition (ACPR). IEEE, pp 141–145
58. Yang J, Lei Z, Liao S, Li SZ (2013) Face liveness detection with component dependent descriptor. ICB 1:2
59. Yang J, Lei Z, Li SZ (2014).Learn convolutional neural network for face anti-spoofing. arXiv:1408.5601
60. Zhang Z, Yan J, Liu S, Lei Z, Yi D, Li SZ (2012). A face antispoofing database with diverse attacks. In: 2012 5th IAPR international conference on biometrics (ICB). IEEE, pp 26–31