Cairo University, Faculty of Computers
and Artificial Intelligence.

Facial Recognition System with Anti-spoofing measures and Liveness Detection Models.

Name	ID
Ahmed Mohammed Abdel-Rashied	20180028
Ahmed Rushdi Mohammed	20180008
Aoss Majed Sultan Zaid	20180432
Eslam Nasser Abdelqader	20180047
Mohammed Walid Moahmmed	20180244
Moahmmed ElSayed AbdelHamid	20180217

supervised by:

dr. M. Wahby

July 2022

Table of Contents

1. Introduction

1.1. Problem definition

Our project is a facial recognition system that overcomes a lot of shortcomings with common facial recognition systems.

According to a survey (The Biometric Survey, 2021), 55% of respondents said that their company is using biometrics today. Of them:

- 69% use the technology for authentication.
- 75% use it for identity verification.
- 31% for fraud detection.

As such, a great percentage of businesses and facilities use some form of check-in system to register the arrival of employees/students/facility members.

Another survey(Peter, 2018) revealed as Shown in Figure 1:

- That fingerprint scanning is the most common form of biometric authentication in the workplace, with 57% of organizations reporting use of it in some capacity.
- The second most popular biometric authentication method in the workplace is facial recognition technology, which is used in 14% of organizations.

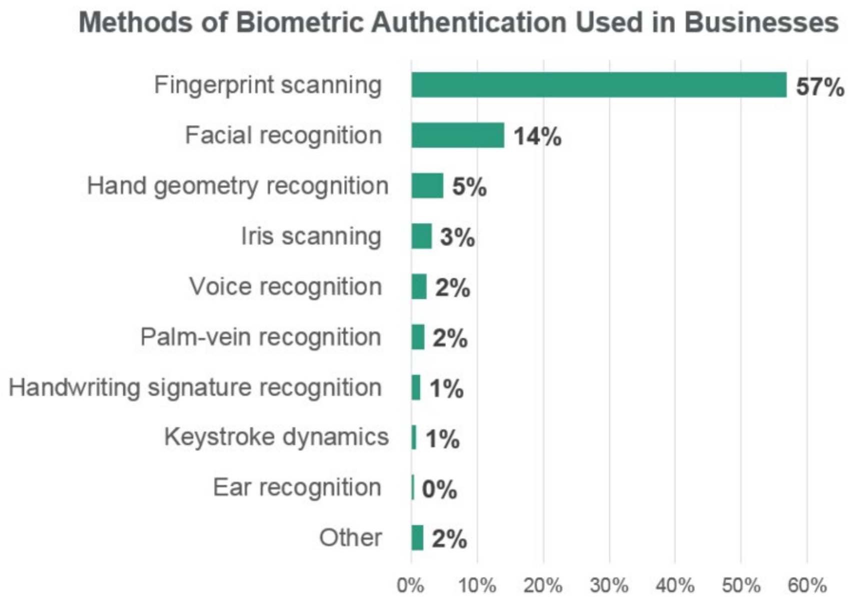


Figure 1. Data by Spiceworks. Surveyed from 500 IT professionals

The most used system (touch id) is inconvenient for many scenarios like:

- Poor Hygiene: in hospitals and clinics or case of global pandemics.
- Work Gloves/dirt on fingers can also obstruct the system from successfully identifying a fingerprint.
- Deformed fingerprint due to working with chemicals or other working conditions.

1.2. Shortcomings of face ID:

Authenticating the user with facial biometrics is a highly demanding biometric modality as face maybe acquired remotely. Such biometric data can be stolen or duplicated by impostors. Most of the existing biometric systems fail to overcome the fraud and identity theft.

Anti-spoofing in biometrics, specifically in face modality, is a critical security problem. A spoof attack in its simplest form consists of an impostor posing a photo of the person to be spoofed. Several measures to detect such attempts like eye blinking detection, depth information can be used.

Other than a simple photo attack, three main kinds of attacks as shown in Figure 2 are (a) warped photo attack where a photo is warped around the face with some movement, (b) cut photo attack where along with warping the photo around the face, the region around the eyes is cut out, and (c) the electronic screen attack or the replay attack where an electronic device is used to play the video of a genuine person.

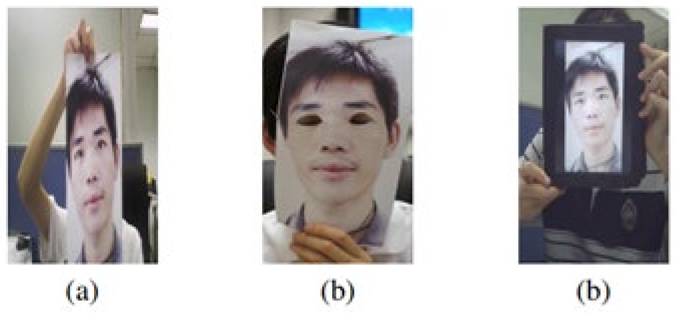


Figure 2 . Examples for three categories of attack: (a) Warped photo attack (b) cut photo attack (c) video attack

These methods are more sophisticated in the fact that they target specific aspects of liveness and replicate it. Although a great amount of work has been done in the field of spoofing detection and many advances have been reached, attacking methodologies have also evolved becoming more and more sophisticated. It is a challenging task to create effective counter measures to detect new realistic and more sophisticated spoofing attacks. Therefore, instead of designing an anti-spoof system that is attack specific, we propose a multi-stage system using deep learning to verify multiple aspects of a person's liveness.

2. Related works

Face recognition techniques have gained widespread attention from biometric communities in the recent decade. Along with other ubiquitous biometric recognition, such as fingerprint, iris, and palm, face recognition has been a dominant mode of biometric identification for authentication and security purposes. The advantages of using face biometric-based systems for access control in various electronic applications are its convenient use, user-friendliness, fast response, cleanliness, and involve minimal human interaction with the device. Without face anti-spoofing (also known as face liveness detection) support, face recognition systems are vulnerable to varieties of face spoofing attacks, also known as face Presentation Attacks (PA). These face PA can be easily generated to gain illegal access to the user device such as cellphone, computer, or intangible assets such as bank accounts. Face PA can be classified into three main types: printed photo PA, replay-video PA, and face-mask PA. While face-mask PA is not readily available because of higher production cost, the former two face PA can be easily generated using high-definition photography or videography. Due to the availability of high-end cameras and printers, and the easy access to social media platforms like Twitter, Facebook, Instagram, WeChat, and YouTube, it has become easier to obtain a photograph and a video of a person's face for face PA production. As a result, face anti-spoofing has become indispensable for face recognition based authentication and verification systems.

In recent years, a multitude of techniques has been developed to detect face PA in face recognition systems. These techniques can be grouped into sensor-based face liveness detection systems and software-based face liveness detection systems. The sensor-based face liveness detection systems utilize RGB-D cameras, Near Infrared Imaging (NIR), thermal cameras, and Kinetics to detect liveness cues in the input face image or video, that can help to identify a live face and face PA. Although the performance of these approaches is quite remarkable compared to software-based approaches, their implementation and maintenance costs are high, which limit its use in portable and handheld electronic devices. On the other hand, software-based face liveness detection methods utilize off the shelf cameras for face image capturing and performing face liveness detection. These techniques either exploit hand-crafted features, deep Convolutional Neural Networks (CNN) or a combination of both to perform face liveness detection.

Current state-of-the-art software-based face liveness detection techniques that operate on dual cameras exploit the disparity or depth information in the input face image or video to detect the face PA. Although these methods have shown better performance in intra-database face liveness detection scenarios, their performance degrades in general on unknown face PA, and in adverse conditions.

One common reason for the drop in performance is that these methods train a classifier like Support Vector Machine (SVM) and CNN using the disparity or depth information computed directly from the RGB or grayscale images. Since the disparity or depth data in case of RGB and grayscale image have fixed number of channels, training a CNN in particular directly on this data limits its capability to learn discriminative and generalized feature maps that can help in identifying unknown face PA.

Until recent years, there was a lack of publicly available face anti-spoofing databases. Performance of most algorithms was evaluated on databases not publicly available [6],[15]. Over the years, as the problem of spoofing attacks against face modality captured attention of researchers, the number of publicly available databases gradually increased.

In the IJCB 2011 competition on 2D face anti-spoofing [8], best strategies for spoof detection have been studied and an attempt to benchmark algorithms on the PRINT-ATTACK database [2] has been made. Other public databases include REPLAY-ATTACK [10], CASIA [11], NUAA PI [12], YALE-RECAPTURED [13], MSU-MFSD [14]. Among these, the REPLAY-ATTACK is probably the most significant one, for its size, multiple and well-defined protocols and attacks covered. It was used in the last edition of the Competition on Countermeasures to 2D Facial Spoofing Attacks held in 2013 [8]. CASIA, REPLAY-ATTACK and MSU-MFSD databases are described in detail in Section 3.

Existing face anti-spoofing approaches can be categorized into two main groups based on strategies used for attack detection: **texture analysis** and **motion-based**.

Texture analysis methods explore patterns that depict the quality of image data. For example, image blurring or quality deterioration in photographs upon recapturing. In [16], Local Binary Patterns (LBP) are used for the texture features to analyze unnatural patterns in spoof samples. However, the technique is specific to just photos and does not address video based or electronic screen attack. [17] use moire patterns generated as a result of recapture of videos and photos to discriminate the live face vs a spoof face in a video replay attack scenario only. Pereira et al in [18] use Local Binary Patterns from Three Orthogonal Planes (LBP-TOP) to explore spatio-temporal texture patterns, which outperformed the LBP-based approach in [16]. The method includes analyzing the motion patterns to distinguish a spoof attack but with masked attacks, human like head motions can be easily replicated.

Motion-based methods use movement information such as eye blinking and lip movement to distinguish a real face from spoof. For example, [19] explores conditional random field to model different stages of eye blinking. The underlying assumption is that real faces show different motion patterns compared to a spoof one. Bao et al. [6] present a countermeasure using optical flow fields that estimate the difference between 2D photograph attacks and 3D real faces.

Such motion-based methods could fail in case of warped photo attack, cut photo attack and masked attack. Several other saliency-based methods [20] use attributes to differentiate between the 3D real face and a 2D fake face for spoofing detection. However advanced masked attacks can pose a fake face as a 3D real face and can be very challenging for the salience detection schemes to detect.

In [7], frequency analysis is performed to detect a photo attack. The algorithm is proposed based on two main assumptions that the spoof photo is always smaller than a real face and the photo is always flat. Therefore, the frequency components of a real face are higher than the spoof photo because of variations of expressions and movement. The two assumptions however do not hold in case of CASIA and REPLAY ATTACK datasets where even the spoof photos are of same size as the real face. Further, video attacks and warping eliminate the possibility of detecting liveness based on the flatness conditions.

Liveness detection methods exploit signs of life from faces of the users that cannot be detected in spoof attempts. In [21], it is shown that a combination of spatial and temporal processing of videos can amplify subtle variations in the faces sufficient to detect liveness. They propose a multiscale approach, Eulerian-based method, to magnify motion without feature tracking or motion estimation.

Texture analysis is the most common approach used. In addition, some works have combined different concepts together. In [22], texture, motion and liveness features are combined together to detect spoofing attacks on PRINTATTACK database.

Most of the above anti-spoofing techniques concentrate on a particular category of attacks and thus the information on the kind of attack is to be inferred in prior. In realistic systems the nature of attack posed is unknown. A robust liveness detection system must be able to address all or most of the existing attacks as well as any new kind of attack that might be used for spoofing. An anti-spoofing algorithm based on Haralick texture features proposed in [23] addresses 3D masks, print attacks and replay attacks but lacks the results of cross-database experiments. In a pulse-based approach used by Team IGD in [24] an Eulerian-based method [21] is used to magnify the spatial and temporal changes. The entire video sequence is reconstructed with magnified motion and color attributes. Two final features, one with the frame representing the highest magnitude, and the other feature representing the average of all frames is fed into AdaBoost classifiers. However, such a representation is not very expressive and computationally expensive.

In [26], instead of using hand-crafted features, a CNN is used to extract patterns and learn deep representations for face liveness detection. This is the first attempt to exploit a deep network to capture discriminative cues between real and fake faces. However, due to the diversity of spoof attacks, their way of spatially augmenting the data might not work well with datasets that have no boundary cues. So, it becomes necessary to have a pre-processing step prior to feeding data to CNN.

In [28], Lakshminarayan et al proposed a system to boost the performance of a face anti-spoofing system by fusing pulse-based features with other spatial and temporal information that markedly define liveness. Thus, developing a spatio-temporal mapping of face and then using a deep Convolutional Neural Network (CNN) to learn discriminative features for liveness detection.

In [27], Rahman et al proposed to supervise a CNN classifier by introducing a disparity layer within CNN to learn the dynamic disparity-maps. Subsequently, the rest of the convolutional layers, following the disparity layer, in the CNN are supervised using the learned dynamic disparity-maps for face liveness detection. Showing that supervising a deep CNN classifier by learning disparity features using the existing CNN layers improves the performance and robustness of CNN to unknown types of face presentation attacks.

In [29], Lin et al proposed a generalized method exploiting both remote photoplethysmography (rPPG) and texture features for face anti-spoofing task. First, multi-scale long-term statistical spectral (MS-LTSS) features with variant granularities are designed for representation of rPPG information. Second, a contextual patch-based convolutional neural network (CP-CNN) is used for extracting global-local and multi-level deep texture features simultaneously. Finally, weight summation strategy is employed for decision level fusion, which helps to generalize the method for not only print attack and replay attack but also mask attack.

3. Face anti-spoofing databases

For our experiments, we use two public face anti-spoofing databases that comprise many attacking scenarios described earlier, CASIA and MSU-MFSD.

3.1. CASIA face anti-spoofing

The CASIA face anti-spoofing database CASAI-FASD contains short videos of both real and spoofing attack attempt of 50 subjects, divided into a train and a test set. Samples were acquired with three devices with different qualities: (i) low quality, with an old 640x480 USB web camera; (ii) normal quality, with a modern 480x 640 USB web camera; and (iii) high quality, using a 1920 x1080 Sony NEX-5 high-definition camera. Three different attacks are considered in this database: (i) warped: curved hard copies of digital photographs of genuine subjects are used to gain illegal access; (ii) cut: the attacks are performed as before, but the eye region in the hard copies have been cut out and the face of the attacker is placed behind, to forge eye blinking; (iii) video: using an iPad, videos of the genuine users are replayed. Hence, each subject has 12 sequences, 3 genuine and 9 fake ones. The overall number of sequences in the database is 600.

3.2. MSU-MFSD

The public available MSU MFSD Database for face spoof attack consists of 280 video clips of photo and video attack attempts to 35 clients. This Database was produced at the Michigan State University Pattern Recognition and Image Processing (PRIP) Lab, in East Lansing, US.

Two types of cameras were used in collecting this database:

1) built-in camera in MacBook Air 13¡± (640x480);

2) front-facing camera in the Google Nexus 5 Android phone (720x480).

Note that the Nexus 5 phone is also equipped with face unlock and anti-spoofing functionalities using its front-facing camera. Therefore, the cameras used in MSU MFSD provide a good simulation to realistic scenario of face spoof detection application on mobile devices.

3.3. Our own dataset generated from YouTube.

The original video [5] contains 21 minutes of different subjects looking at the camera. The video was split according to subject in python using OpenCV and manually acquired time stamps of each subject.

The attack data was captured using a Huawei Y7 (2019) front facing camera with the video replay on a Redmi note 11 AMOLED screen. The data was then split in DaVinci Resolve according to subject and then further split into 7 to 10 second clips.

4. Models for liveness detection

We Studied two different types of models: (i) single frame classification, (ii) video classification. Which will be discussed in detail in this section.

We will first discuss single frame classification.

4.1. Frame Classification: Face Antispoofing Method Using Color Texture Segmentation

we propose a liveness face detection method based on the color and texture information of a face image. The proposed method analyzes the combined color-texture information in terms of its luminance and color difference channels using an LBP descriptor. For color-texture information analysis, the Cb, S, and H bands are used from the color spaces [30].

Why don't we use RGB?

RGB is a color space commonly used for sensing and displaying color images. However, its use in image analysis is typically limited because the three colors (red, green, and blue) are not separated according to luminance and color difference information. Thus, it is common to additionally convert the RGB information into YCbCr and HSV information before use. These two latter color spaces are based on luminance and chrominance information [31–34]. In particular, the YCbCr Color space separates RGB into luminance (Y), chrominance blue (Cb), and chrominance red (Cr). Similarly, the HSV color space uses the hue(H) and saturation(S) dimensions to define the color differences of the image, and the value(V) dimension corresponds to the luminance. These color space equations are in Figure 3

$$V = \max(R, G, B),$$

$$S = \begin{cases} \dfrac{V - \min(R, G, B)}{V}, & \text{if } V \neq 0, \\ 0, & \text{if } V = 0, \end{cases}$$

$$H = \begin{cases} \dfrac{60(G - B)}{V - \min(R, G, B)}, & \text{if } V = R, \\ 120 + \dfrac{60(B - R)}{V - \min(R, G, B)}, & \text{if } V = G, \\ 240 + \dfrac{60(R - G)}{V - \min(R, G, B)}, & \text{if } V = B, \end{cases}$$

if $H < 0, H = H + 360.$

The YCbCr calculation formula is shown as

$$Y = 0.299R + 0.587G + 0.114B,$$
$$\text{Cb} = 0.564(B - Y),$$
$$\text{Cr} = 0.713(R - Y).$$

Figure3 . color space equations

What are the components of color spaces?

The RGB color space contains three color components, red, green, and blue; the YCbCr color space contains brightness and saturation information, and the HSV color space contains three components: hue, saturation, and brightness. Each color space contains different information and has its own characteristics. RGB contains rich spatial information that most closely resembles the colors seen by humans, while the YCbCr and HSV color spaces contain information that is more sensitive to brightness. The RGB color space can be converted into HSV and YCbCr [30].

What is the feature for images?

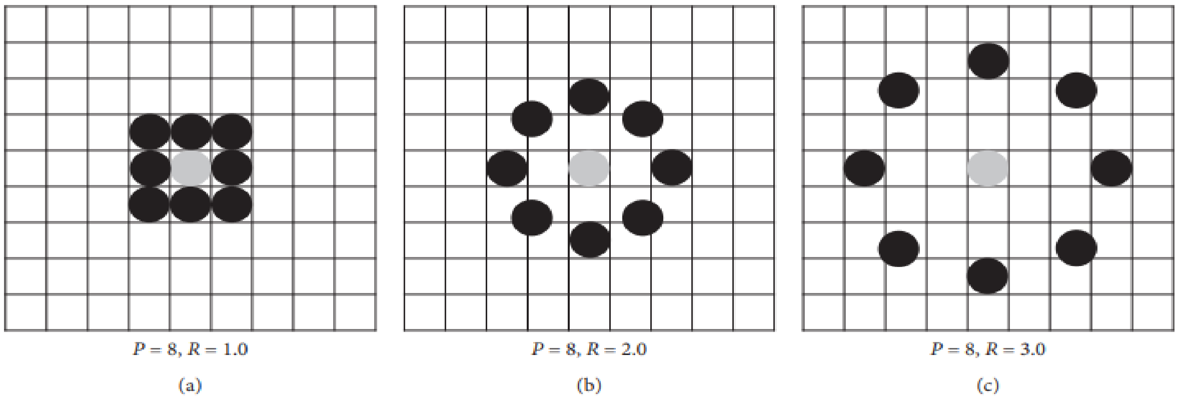
In our model, we only use color texture information that is analyzed using Cb, S, and V bands in the color spaces. Then we use a descriptor which is the LBP (Local Binary Pattern). LBPs [35, 36] are a feature developed for classifying image textures. Since then, LBPs have been used for face recognition. LBPs are a simple operation used for image analysis and recognition and are robust to changes in discrimination and lighting. The LBP equation is in Figure 4:

Here, gp ranges over the pixel values excluding the center pixel and gc is the center pixel in equation. P is the number of adjacent pixels and R is the radius of the circle.

$$\mathrm{LBP}(p, r) = \sum_{p=1}^{P-1} s(g_p - g_c) 2^P,$$

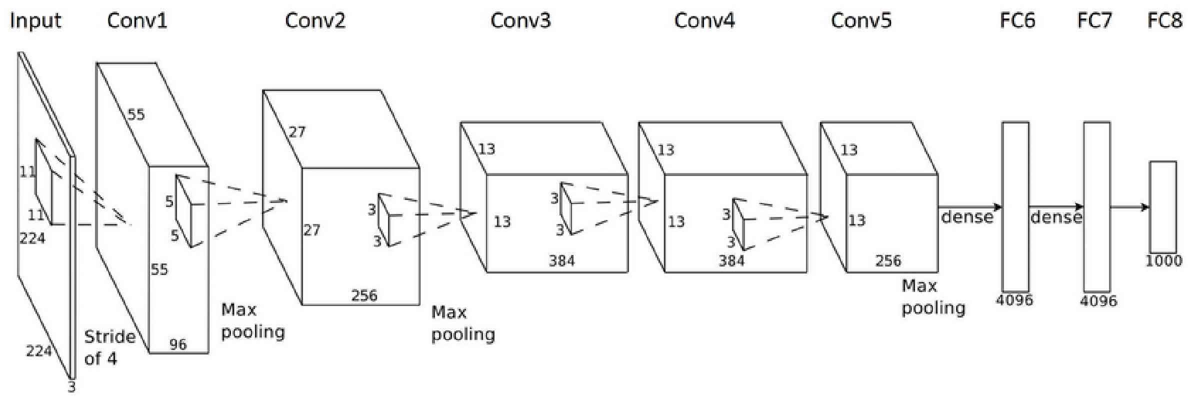
$$s(x) = \begin{cases} 1, & \text{if } x \geq 0, \\ 0, & \text{otherwise.} \end{cases}$$

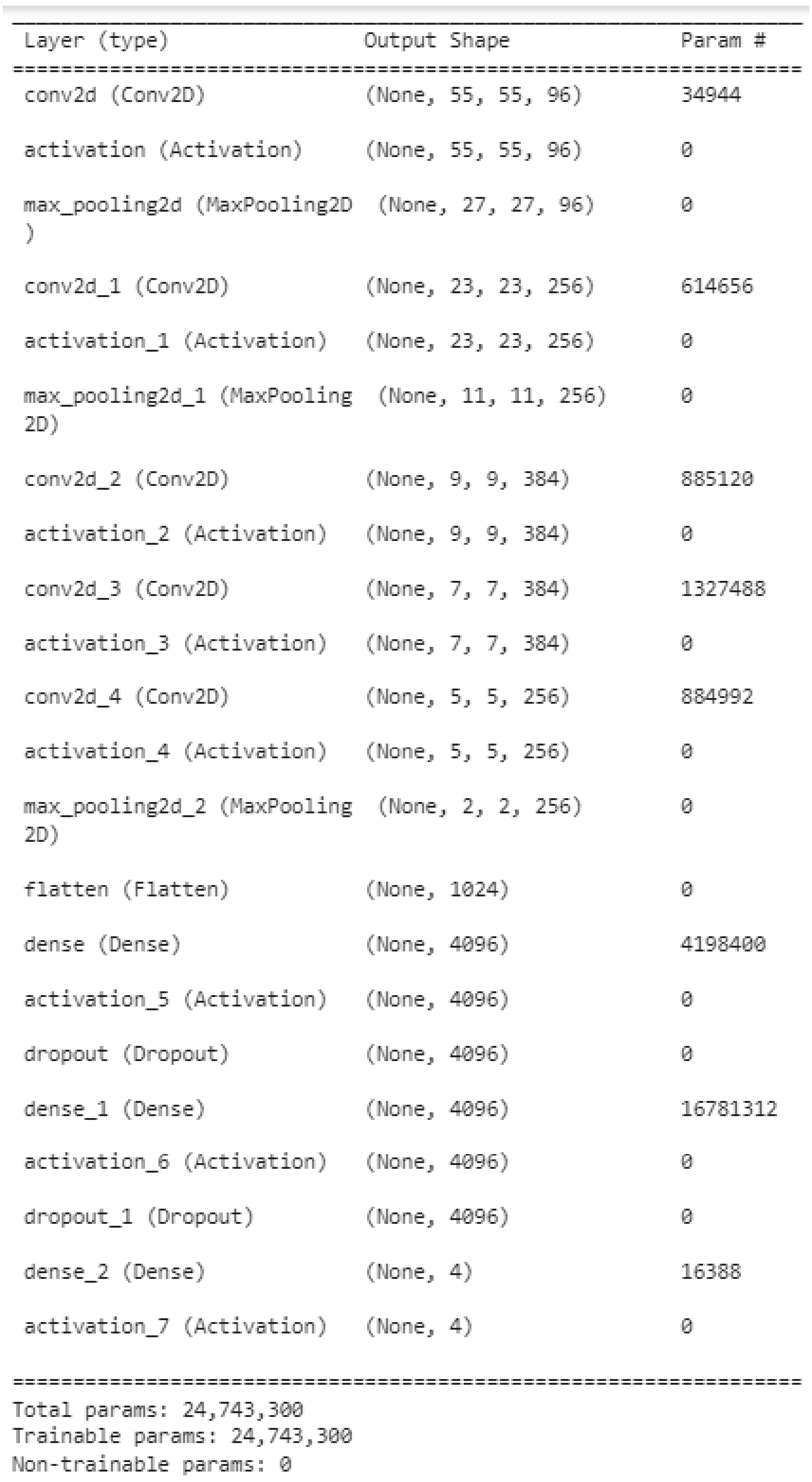
Figure4 . LBP Equation



CNN Model:

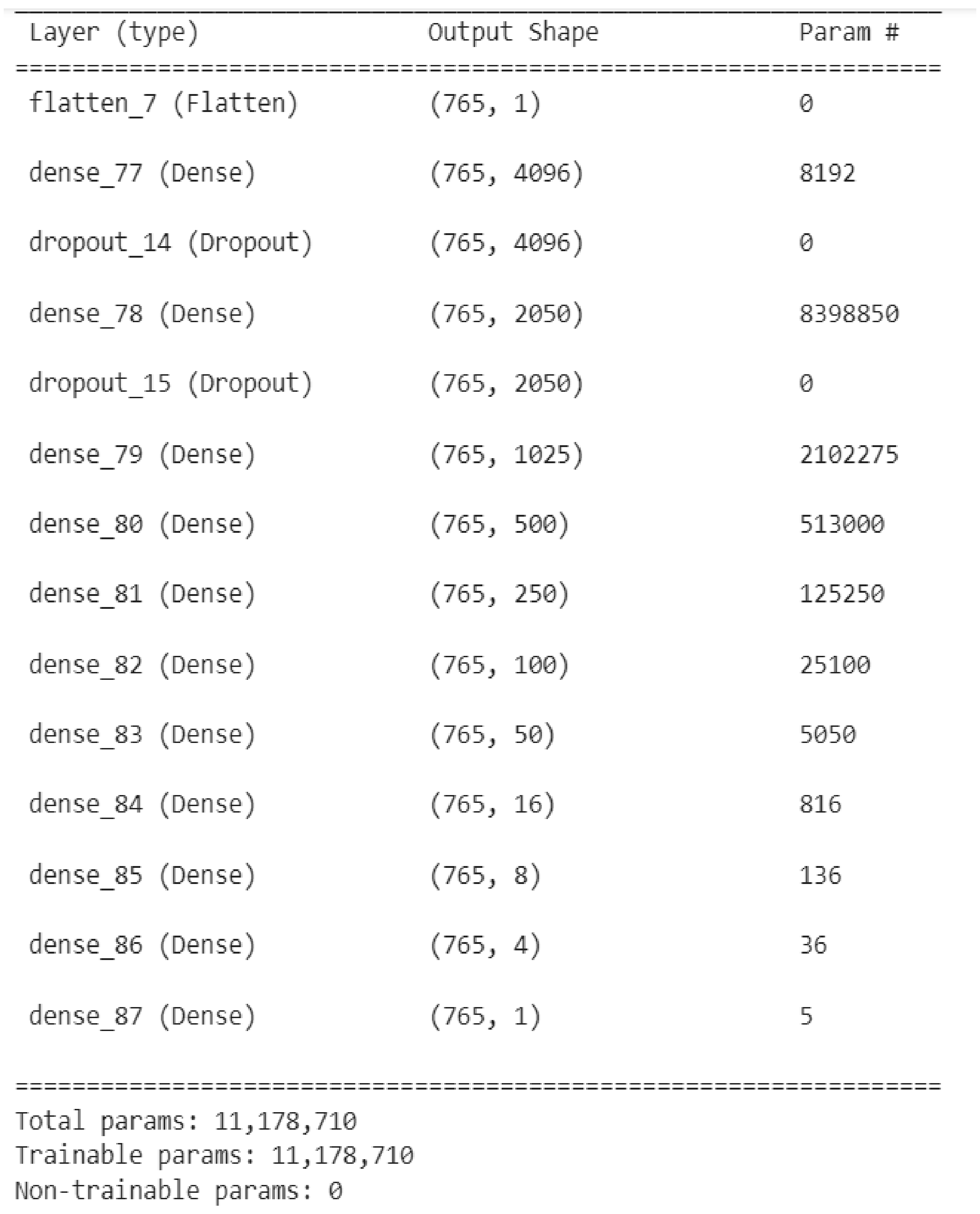
After we make the LBP for H, S and Cb, we merge the three LBPs for H, S, and Cb, until we have one image that we can then feed into the model. We use Alex Net like architecture based on CNN. Alex Net is a basic model utilizing a convolutional layer, a pooling layer, and a fully connected layer [37]. Alex Net consists of five convolution layers and three full-connected (FC) layers, where the last FC layer uses SoftMax as an active function for category classification. Alex Net CNN architecture:





```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 55, 55, 96)        34944

 activation (Activation)     (None, 55, 55, 96)        0

 max_pooling2d (MaxPooling2D  (None, 27, 27, 96)       0
 )

 conv2d_1 (Conv2D)           (None, 23, 23, 256)       614656

 activation_1 (Activation)   (None, 23, 23, 256)       0

 max_pooling2d_1 (MaxPooling  (None, 11, 11, 256)      0
 2D)

 conv2d_2 (Conv2D)           (None, 9, 9, 384)         885120

 activation_2 (Activation)   (None, 9, 9, 384)         0

 conv2d_3 (Conv2D)           (None, 7, 7, 384)         1327488

 activation_3 (Activation)   (None, 7, 7, 384)         0

 conv2d_4 (Conv2D)           (None, 5, 5, 256)         884992

 activation_4 (Activation)   (None, 5, 5, 256)         0

 max_pooling2d_2 (MaxPooling  (None, 2, 2, 256)        0
 2D)

 flatten (Flatten)           (None, 1024)              0

 dense (Dense)               (None, 4096)              4198400

 activation_5 (Activation)   (None, 4096)              0

 dropout (Dropout)           (None, 4096)              0

 dense_1 (Dense)             (None, 4096)              16781312

 activation_6 (Activation)   (None, 4096)              0

 dropout_1 (Dropout)         (None, 4096)              0

 dense_2 (Dense)             (None, 4)                 16388

 activation_7 (Activation)   (None, 4)                 0

=================================================================
Total params: 24,743,300
Trainable params: 24,743,300
Non-trainable params: 0
```

Deep Neural Network Model:
After calculating the LBP for H, S and Cb, we calculate the histograms for each channel of the LBPs for H, S, and Cb and concatenate them together together, we can then feed the data into the fully connected model. We used fully connected Encoder like architecture with dropout layers. The Full Neural Network architecture:



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_7 (Flatten)         (765, 1)                  0

 dense_77 (Dense)            (765, 4096)               8192

 dropout_14 (Dropout)        (765, 4096)               0

 dense_78 (Dense)            (765, 2050)               8398850

 dropout_15 (Dropout)        (765, 2050)               0

 dense_79 (Dense)            (765, 1025)               2102275

 dense_80 (Dense)            (765, 500)                513000

 dense_81 (Dense)            (765, 250)                125250

 dense_82 (Dense)            (765, 100)                25100

 dense_83 (Dense)            (765, 50)                 5050

 dense_84 (Dense)            (765, 16)                 816

 dense_85 (Dense)            (765, 8)                  136

 dense_86 (Dense)            (765, 4)                  36

 dense_87 (Dense)            (765, 1)                  5

=================================================================
Total params: 11,178,710
Trainable params: 11,178,710
Non-trainable params: 0
```

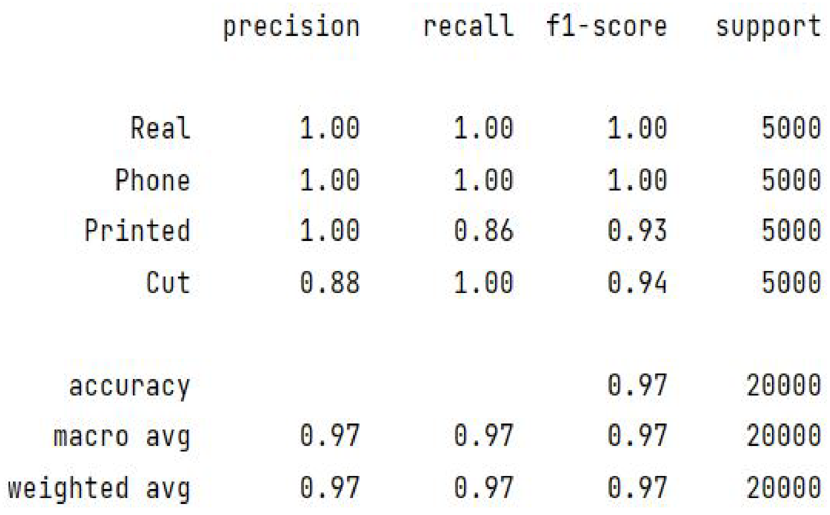
Experiments

For the **CNN Model** We've run our experiment on the CASIA-FASD and MSU-MSFD datasets and with experimentation we found the best optimizer is Stochastic gradient descent (SGD) with Learning Rate = 0.0001 momentum = 0.001 with Loss Function Categorical Cross entropy, and hyperparameters epochs=50, batch size=265.

For the **NN Model** We've run our experiment on the CASIA-FASD and MSU-MSFD datasets and with experimentation we found the best optimizer is Adam with Learning Rate=0.001 with Loss Function Binary Crossentropy, and hyperparameters epochs=20, batch size=500.

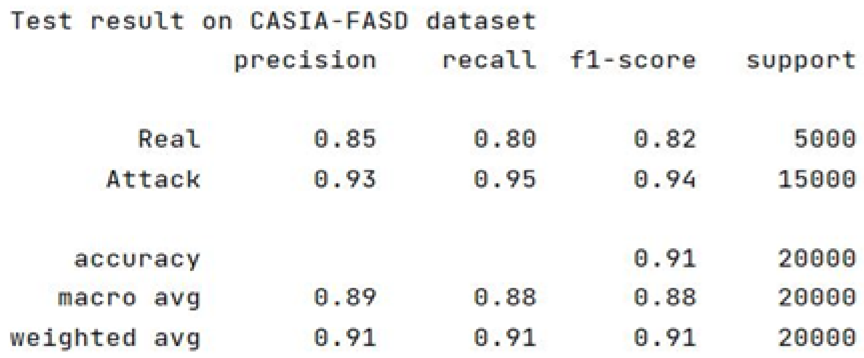
Results

The results of **CNN model** were very good as it scored on training, an accuracy rate = 99.5%, and when tested on the two data, it scored an accuracy rate = 97%.



```
              precision    recall  f1-score   support

        Real       1.00      1.00      1.00      5000
       Phone       1.00      1.00      1.00      5000
     Printed       1.00      0.86      0.93      5000
         Cut       0.88      1.00      0.94      5000

    accuracy                           0.97     20000
   macro avg       0.97      0.97      0.97     20000
weighted avg       0.97      0.97      0.97     20000
```

The results of **DNN model** were tested on the CASIA-FASD data set, it scored an accuracy rate = 91%



```
Test result on CASIA-FASD dataset
              precision    recall  f1-score   support

        Real       0.85      0.80      0.82      5000
      Attack       0.93      0.95      0.94     15000

    accuracy                           0.91     20000
   macro avg       0.89      0.88      0.88     20000
weighted avg       0.91      0.91      0.91     20000
```

4.2. Video classification:

First, Volumetric CNN:

Our first approach to video classification was using a volumetrics CNN that takes 24 frames of video each resized to 100 x 100 and converted to grayscale.

Face Localization is used on each frame of the video. The bounding boxes of the face are then expanded by a factor of 100% of dimensions to include relevant background information. Example in Figure 5.

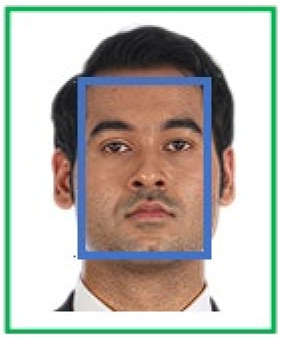


Figure5 . Bounding box expansion

To Compensate for unstable bounding box in video we used a moving average bounding box as follows:

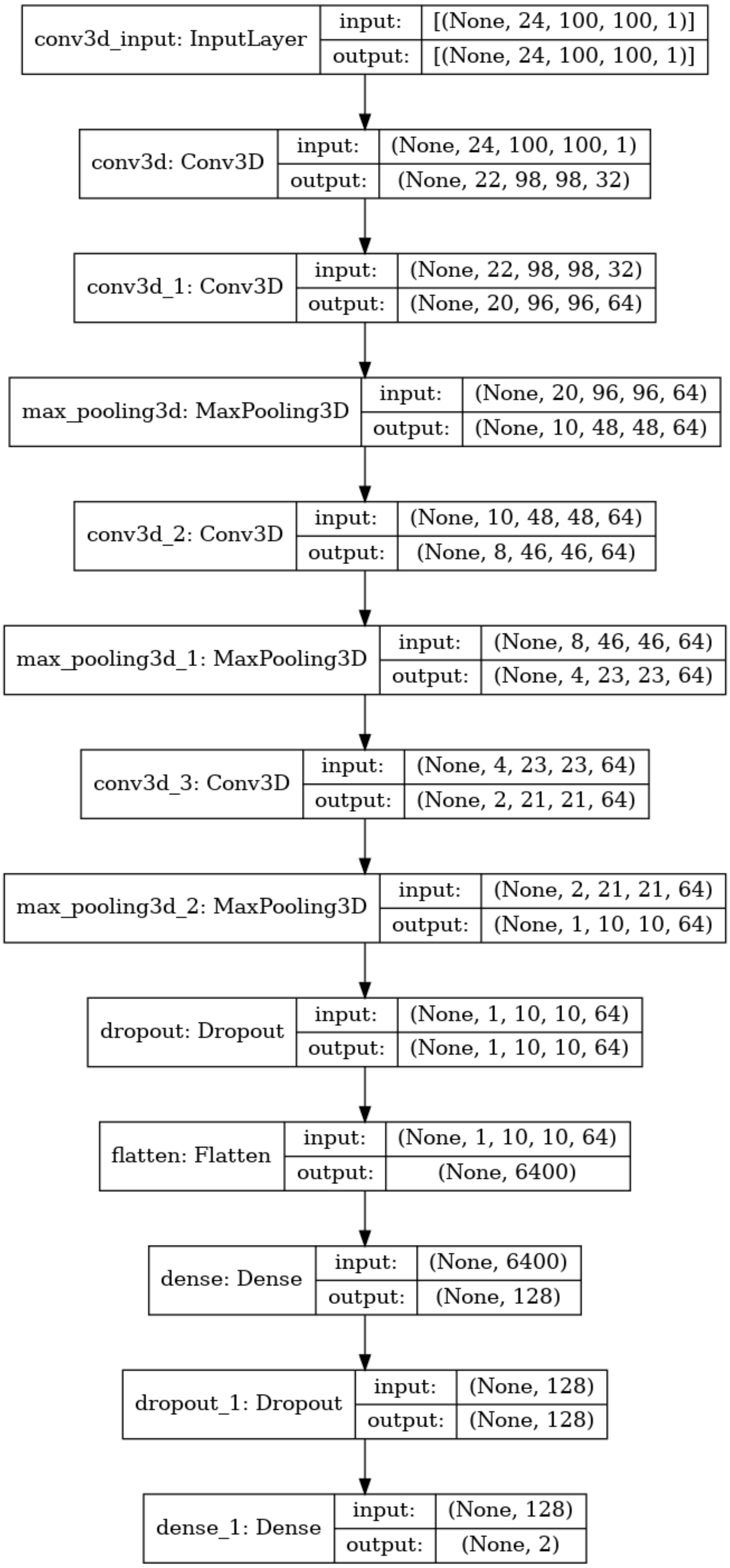
Bbox = 0.2 * new_bbox + (1 - 0.2) prev_bbox

We call this method Stabilized Face detection and from our testing it increases the stability of the obtained region greatly.

Histogram equalization is then preformed after retrieving the region of interest

The data of shape (24,100,100,1) is then fed into the Volumetric CNN model.

The model Architecture:


conv3d_input: InputLayer | input: [(None, 24, 100, 100, 1)] | output: [(None, 24, 100, 100, 1)]
conv3d: Conv3D | input: (None, 24, 100, 100, 1) | output: (None, 22, 98, 98, 32)
conv3d_1: Conv3D | input: (None, 22, 98, 98, 32) | output: (None, 20, 96, 96, 64)
max_pooling3d: MaxPooling3D | input: (None, 20, 96, 96, 64) | output: (None, 10, 48, 48, 64)
conv3d_2: Conv3D | input: (None, 10, 48, 48, 64) | output: (None, 8, 46, 46, 64)
max_pooling3d_1: MaxPooling3D | input: (None, 8, 46, 46, 64) | output: (None, 4, 23, 23, 64)
conv3d_3: Conv3D | input: (None, 4, 23, 23, 64) | output: (None, 2, 21, 21, 64)
max_pooling3d_2: MaxPooling3D | input: (None, 2, 21, 21, 64) | output: (None, 1, 10, 10, 64)
dropout: Dropout | input: (None, 1, 10, 10, 64) | output: (None, 1, 10, 10, 64)
flatten: Flatten | input: (None, 1, 10, 10, 64) | output: (None, 6400)
dense: Dense | input: (None, 6400) | output: (None, 128)
dropout_1: Dropout | input: (None, 128) | output: (None, 128)
dense_1: Dense | input: (None, 128) | output: (None, 2)

Experiments

The model was trained on the train and tuned on the test portion of the CASIA Database.

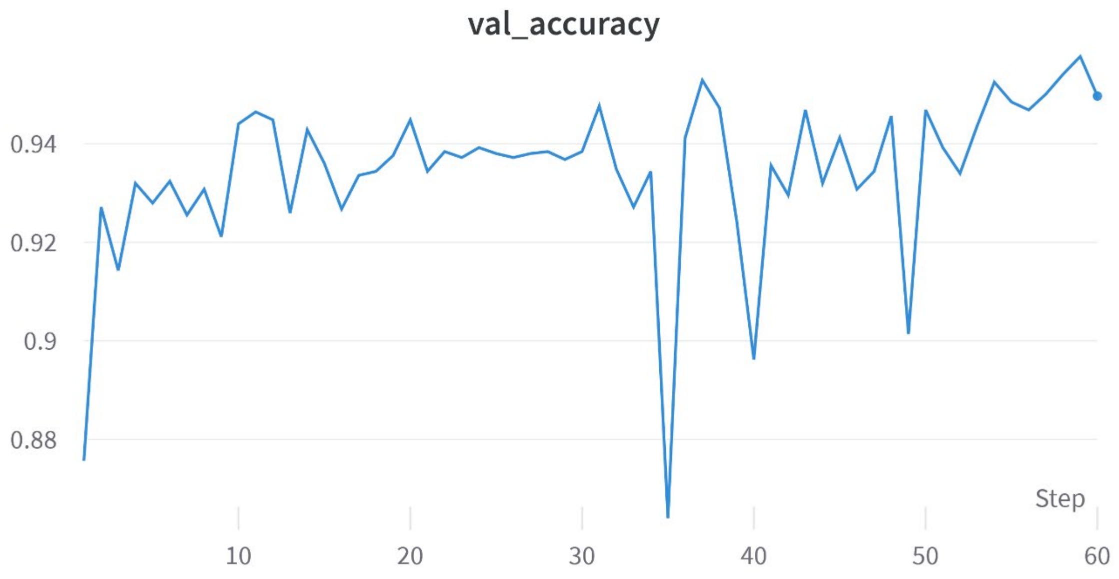
After a lot of experimentation, we used batch size of 16, Adam optimizer initial learning rate of 0.001, and a total of 60 epochs.

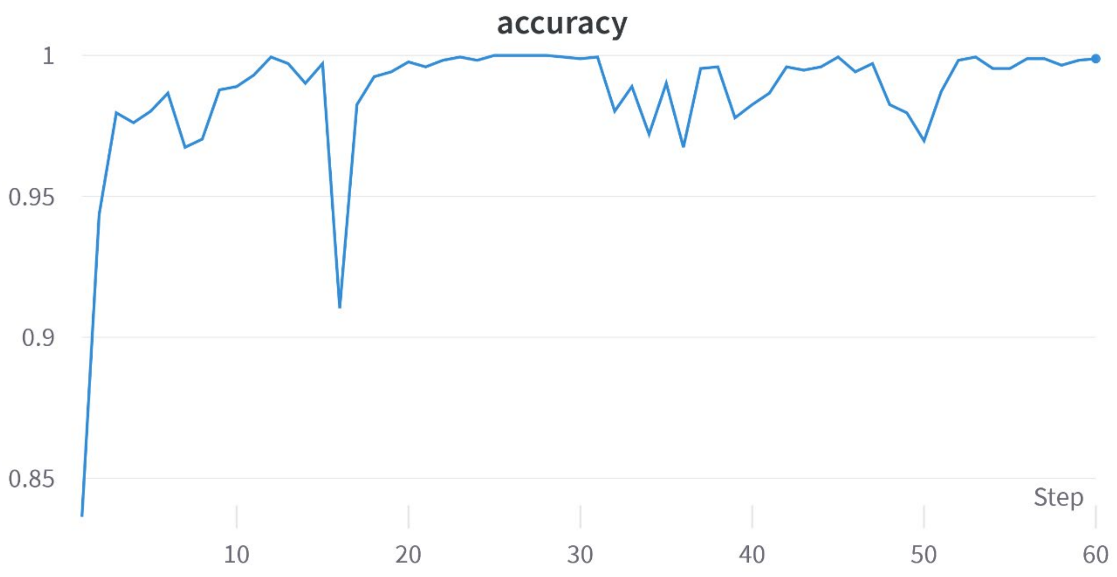
We also used model checkpoint callback and learning rate reduction on plateau with parameters as follows:

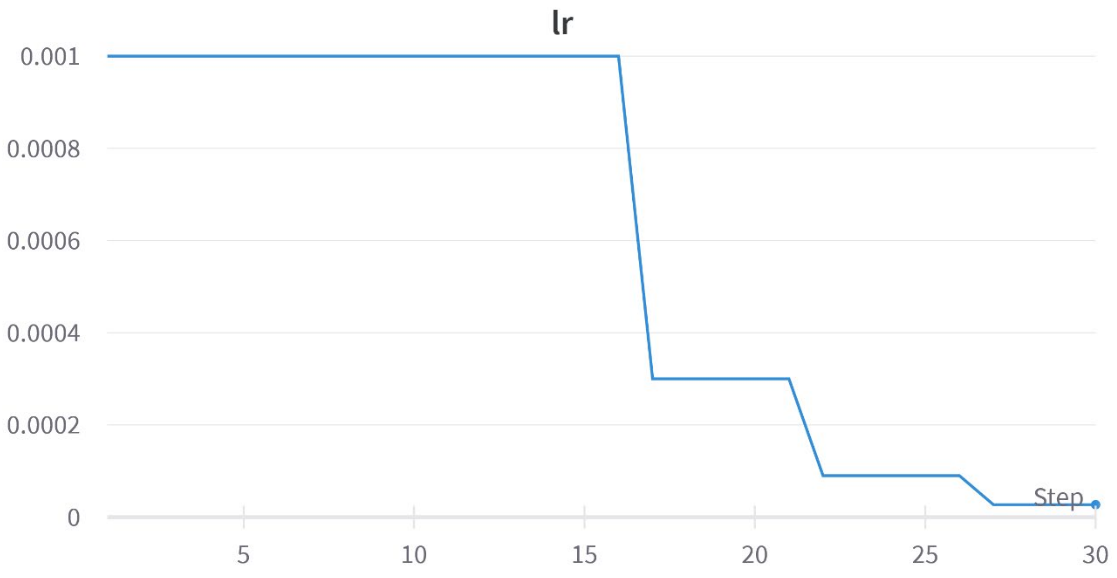
```
checkpoint_filepath = 'bestmodel'
model_checkpoint_callback = keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_filepath,
    save_weights_only=True,
    monitor='val_accuracy',
    verbose=1,
    mode='max',
    save_best_only=True)


lr_red = keras.callbacks.ReduceLROnPlateau(
    monitor='val_accuracy',
    factor=0.3,
    patience=5,
    verbose=1,
    mode='max',
    min_lr=1e-5
)
```

We then train the model for 30 epochs then load the best weights and retrain the model for another 30 epochs with much lower learning rate.

Here are the charts for the training process:



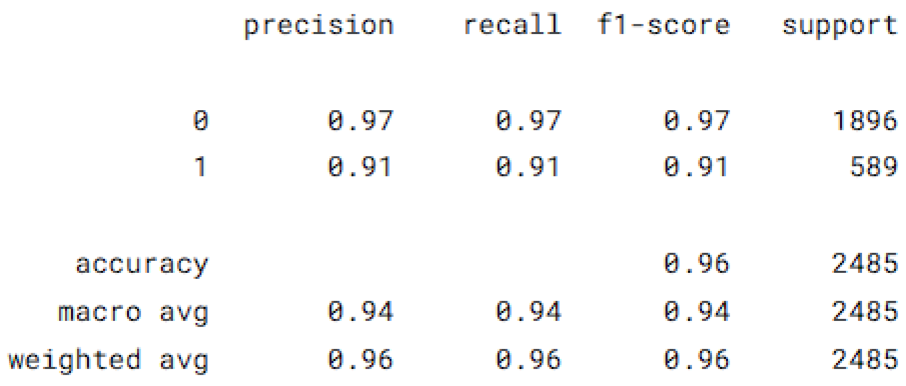

accuracy
1
0.95
0.9
0.85
Step
10
20
30
40
50
60


lr
0.001
0.0008
0.0006
0.0004
0.0002
0
Step
5
10
15
20
25
30

Results:

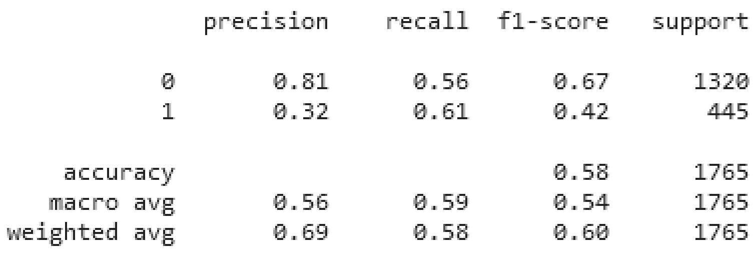
The test results are as follows:

0: Attack, 1: Real



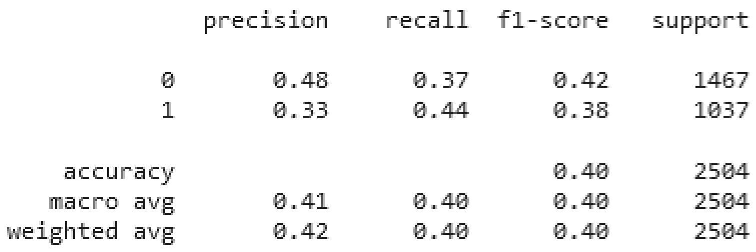
```
              precision    recall  f1-score   support

           0       0.97      0.97      0.97      1896
           1       0.91      0.91      0.91       589

    accuracy                           0.96      2485
   macro avg       0.94      0.94      0.94      2485
weighted avg       0.96      0.96      0.96      2485
```

The model was also Intra-tested on the full MSU-MFSD with the following results:



```
              precision    recall  f1-score   support

           0       0.81      0.56      0.67      1320
           1       0.32      0.61      0.42       445

    accuracy                           0.58      1765
   macro avg       0.56      0.59      0.54      1765
weighted avg       0.69      0.58      0.60      1765
```

The model was also Intra-tested on our generated dataset with the following results:



```
              precision    recall  f1-score   support

           0       0.48      0.37      0.42      1467
           1       0.33      0.44      0.38      1037

    accuracy                           0.40      2504
   macro avg       0.41      0.40      0.40      2504
weighted avg       0.42      0.40      0.40      2504
```

Second, Spatio-temporal Mapping:
This Method was proposed in [28] but no source code was provided as such we followed the written instruction to make this model.

An overview is provided in Figure 6.

First: extract face from each frame using stabilized face detection.

Second: Expand bounding box by a factor of 0.2 and resize frame to 64 x 64.

Third: preform illumination correction on each frame by converting the frames to YUV and preforming histogram equalization on the Lumina Channel (Y) then revert back to the original format (BGR).

Fourth: make an Equiripple band pass filter [0.7, 2.6] Hz constructed with scipy.signal.remez with num taps = 800 and transition width = 0.1.

Fifth: convolve the filter with each pixel channel by channel along the time axis.

Sixth: calculate the ESD (energy spectral density) using scipy.signal.welch with parameter (scaling='spectrum') and summing along each channel.

Seventh: now that every pixel has 3 values E(R), E(G), E(B)

We can calculate E(total)as follows:

$$E = \sqrt{E_R^2 + E_G^2 + E_B^2}$$

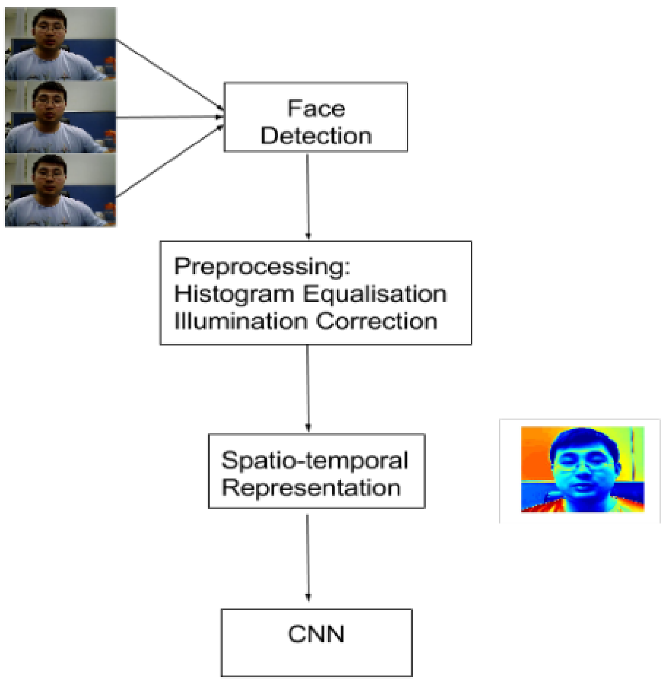
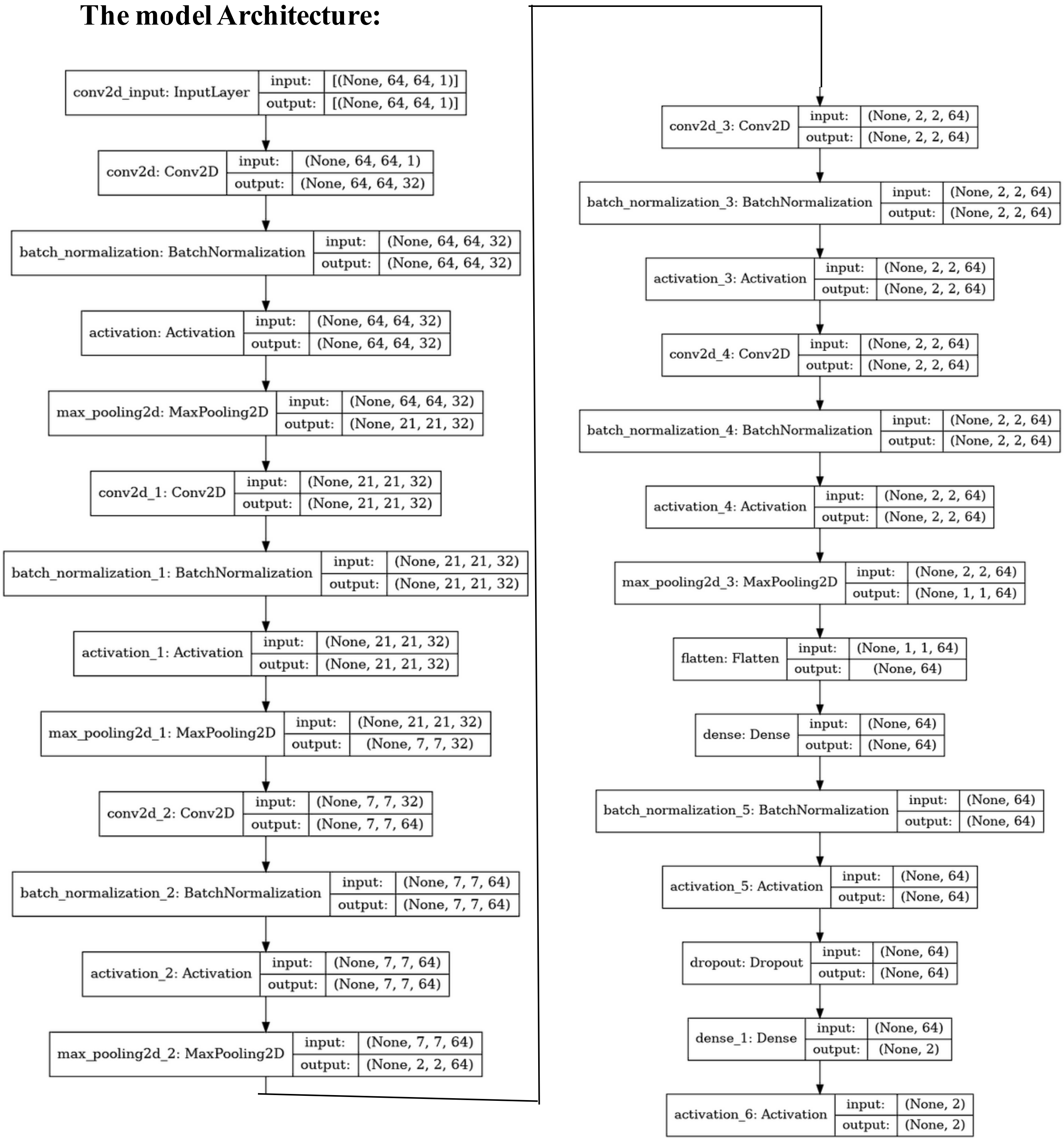

Face Detection
Preprocessing: Histogram Equalisation Illumination Correction
Spatio-temporal Representation
CNN

Figure 6. An overview of the architecture for the proposed system.

The model Architecture:



conv2d_input: InputLayer | input: [(None, 64, 64, 1)] | output: [(None, 64, 64, 1)]
conv2d: Conv2D | input: (None, 64, 64, 1) | output: (None, 64, 64, 32)
batch_normalization: BatchNormalization | input: (None, 64, 64, 32) | output: (None, 64, 64, 32)
activation: Activation | input: (None, 64, 64, 32) | output: (None, 64, 64, 32)
max_pooling2d: MaxPooling2D | input: (None, 64, 64, 32) | output: (None, 21, 21, 32)
conv2d_1: Conv2D | input: (None, 21, 21, 32) | output: (None, 21, 21, 32)
batch_normalization_1: BatchNormalization | input: (None, 21, 21, 32) | output: (None, 21, 21, 32)
activation_1: Activation | input: (None, 21, 21, 32) | output: (None, 21, 21, 32)
max_pooling2d_1: MaxPooling2D | input: (None, 21, 21, 32) | output: (None, 7, 7, 32)
conv2d_2: Conv2D | input: (None, 7, 7, 32) | output: (None, 7, 7, 64)
batch_normalization_2: BatchNormalization | input: (None, 7, 7, 64) | output: (None, 7, 7, 64)
activation_2: Activation | input: (None, 7, 7, 64) | output: (None, 7, 7, 64)
max_pooling2d_2: MaxPooling2D | input: (None, 7, 7, 64) | output: (None, 2, 2, 64)
conv2d_3: Conv2D | input: (None, 2, 2, 64) | output: (None, 2, 2, 64)
batch_normalization_3: BatchNormalization | input: (None, 2, 2, 64) | output: (None, 2, 2, 64)
activation_3: Activation | input: (None, 2, 2, 64) | output: (None, 2, 2, 64)
conv2d_4: Conv2D | input: (None, 2, 2, 64) | output: (None, 2, 2, 64)
batch_normalization_4: BatchNormalization | input: (None, 2, 2, 64) | output: (None, 2, 2, 64)
activation_4: Activation | input: (None, 2, 2, 64) | output: (None, 2, 2, 64)
max_pooling2d_3: MaxPooling2D | input: (None, 2, 2, 64) | output: (None, 1, 1, 64)
flatten: Flatten | input: (None, 1, 1, 64) | output: (None, 64)
dense: Dense | input: (None, 64) | output: (None, 64)
batch_normalization_5: BatchNormalization | input: (None, 64) | output: (None, 64)
activation_5: Activation | input: (None, 64) | output: (None, 64)
dropout: Dropout | input: (None, 64) | output: (None, 64)
dense_1: Dense | input: (None, 64) | output: (None, 2)
activation_6: Activation | input: (None, 2) | output: (None, 2)

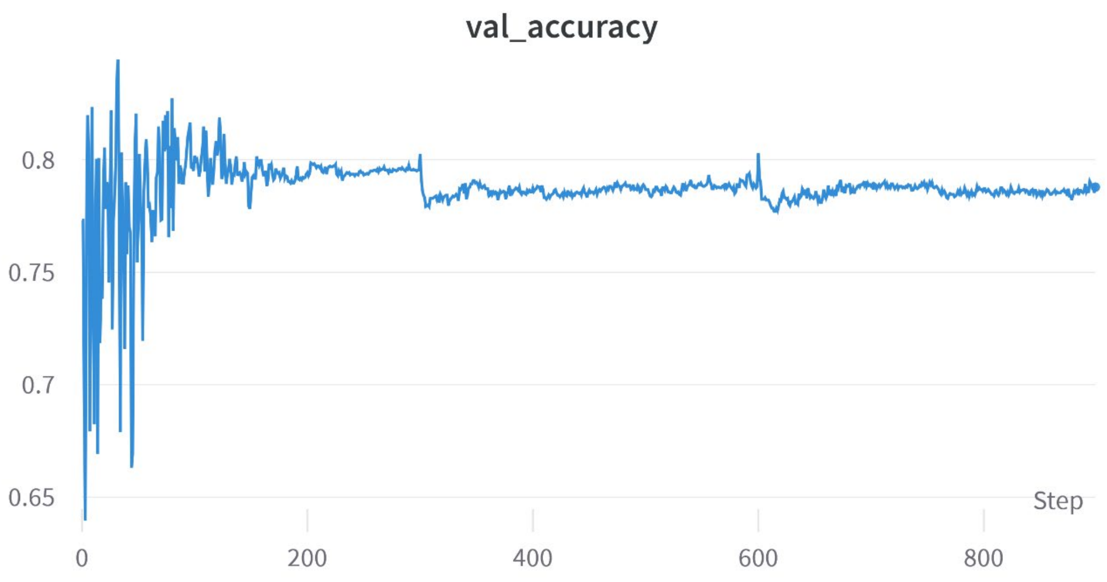
Experiments:
The model was trained on the train portion of CASIA.

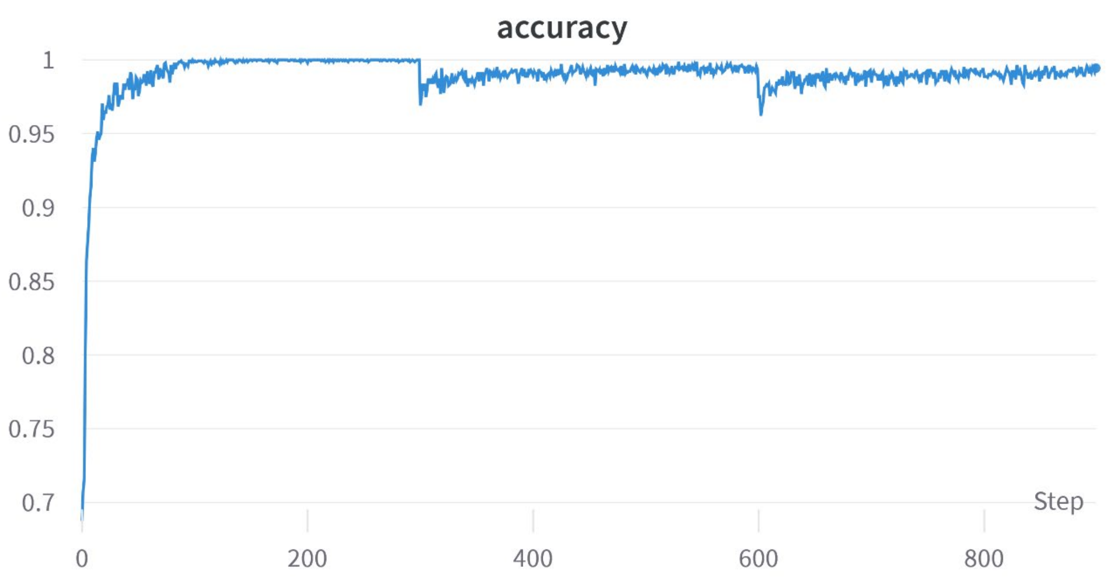
We used the same method of training as the volumetric CNN. We used a batch size of 16, trained for 300 epochs per iteration for 3 iterations, with an initial LR of 0.01 and SGD Optimizer with 0.9 momentum.

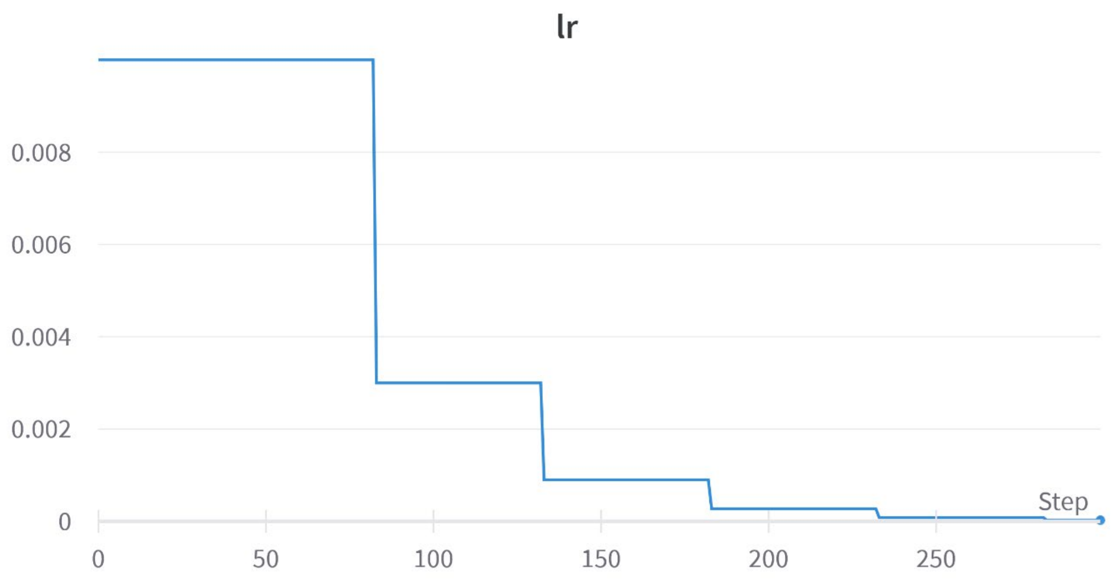
We also Model checkpoint, LR reduction and used data augmentation on training data using keras image generator:

```
aug = keras.preprocessing.image.ImageDataGenerator(
    rotation_range=10,
    fill_mode='reflect',
    horizontal_flip=True,
)

lr_red = keras.callbacks.ReduceLROnPlateau(
    monitor='val_accuracy',
    factor=0.3,
    patience=50,
    verbose=1,
    mode='max',
    min_lr=1e-5,
)

checkpoint_filepath = 'bestmodel'
model_checkpoint_callback = keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_filepath,
    save_weights_only=True,
    monitor='val_accuracy',
    verbose=1,
    mode='max',
    save_best_only=True)
```

Here are the charts for the training process:







Results:
The test results are as follows:

CAISA(Test)

0: Real, 1: Attack

```
              precision    recall  f1-score   support

           0       0.72      0.51      0.60       631
           1       0.87      0.94      0.90      2150

    accuracy                           0.84      2781
   macro avg       0.79      0.73      0.75      2781
weighted avg       0.84      0.84      0.83      2781
```

5. Discussion

Issues with the CASIA-FASD:

1. All replay attacks are performed with the same device (IPAD).
2. All subjects are Asian as such the database lacks diversity.
3. No subjects have beards which resulted in confusion when trying to predict bearded people.

Issues with the CASIA-FASD:

1. No replay attacks using smartphones.
2. Data is more diverse than CASIA.
3. Lack of beards.

Issues with our generated data:

1. The video was filmed in a studio with a dark background and is unrealistic for a real-world scenario
2. The setup used to generate the attack images was of very poor lighting quality and camera quality

6. Full System Overview

Here we will discuss different system components and the verification process.

6.1. FaceNet

Introduction

Face recognition (FR) has been the prominent biometric technique for identity authentication and has been widely used in many areas, such as military, finance, public security and daily life. FR has been a long-standing research topic in the CVPR community. In the early 1990s, the study of FR became popular following the introduction of the historical Eigenface approach [38]. The milestones of feature-based FR over the past years are presented in Fig. 7, in which the times of four major technical streams are highlighted. The holistic approaches derive the low-dimensional representation through certain distribution assumptions, such as linear subspace [39][40][41], manifold [42][43][44], and sparse representation [45][46][47][48]. This idea dominated the FR community in the 1990s and 2000s. However, a well-known problem is that these theoretically plausible holistic methods fail to address the uncontrolled facial changes that deviate from their prior assumptions. In the early 2000s, this problem gave rise to local-feature-based FR. Gabor [49] and LBP [50], as well as their multilevel and high-dimensional extensions [51][52][53], achieved robust performance through some invariant properties of local filtering. Unfortunately, handcrafted features suffered from a lack of distinctiveness and compactness. In the early 2010s, learning-based local descriptors were introduced to the FR community [54][55][56], in which local filters are learned for better distinctiveness and the encoding codebook is learned for better compactness. However, these shallow representations still have an inevitable limitation on robustness against the complex nonlinear facial appearance variations.

In general, traditional methods attempted to recognize human face by one- or two-layer representations, such as filtering responses, histogram of the feature codes, or distribution of the dictionary atoms. The research community studied intensively to separately improve the preprocessing, local descriptors, and feature transformation, but these approaches improved FR accuracy slowly. What's worse, most methods aimed to address one aspect of unconstrained facial changes only, such as lighting, pose, expression, or disguise. There was no any integrated technique to address these unconstrained challenges integrally. As a result, with continuous efforts of more than a decade, "shallow" methods only improved the accuracy of the LFW benchmark to about 95% [52], which indicates that "shallow" methods are insufficient to extract stable identity feature invariant to real-world changes. Due to the insufficiency of this technical, facial recognition systems were often reported with unstable performance or failures with countless false alarms in real-world applications

But all that changed in 2012 when AlexNet won the ImageNet competition by a large margin using a technique called deep learning [59]. Deep learning methods, such as convolutional neural networks, use a cascade of multiple layers of processing units for feature extraction and transformation. They learn multiple levels of representations that correspond to different levels of abstraction. The levels form a hierarchy of concepts, showing strong invariance to the face pose, lighting, and expression changes, as shown in Fig. 8. It can be seen from the figure that the first layer of the deep neural network is somewhat similar to the Gabor feature found by human scientists with years of experience. The second layer learns more complex texture features. The features of the third layer are more complex, and some simple structures have begun to appear, such as high-bridged nose and big eyes. In the fourth, the network output is enough to explain a certain facial attribute, which can make a special response to some clear abstract concepts such as smile, roar, and even blue eye. In conclusion, in deep convolutional neural networks (CNN), the lower layers automatically learn the features similar to Gabor and SIFT designed for years or even decades (such as initial layers in Fig. 8), and the higher layers further learn higher level abstraction. Finally, the combination of these higher-level abstraction represents facial identity with unprecedented stability

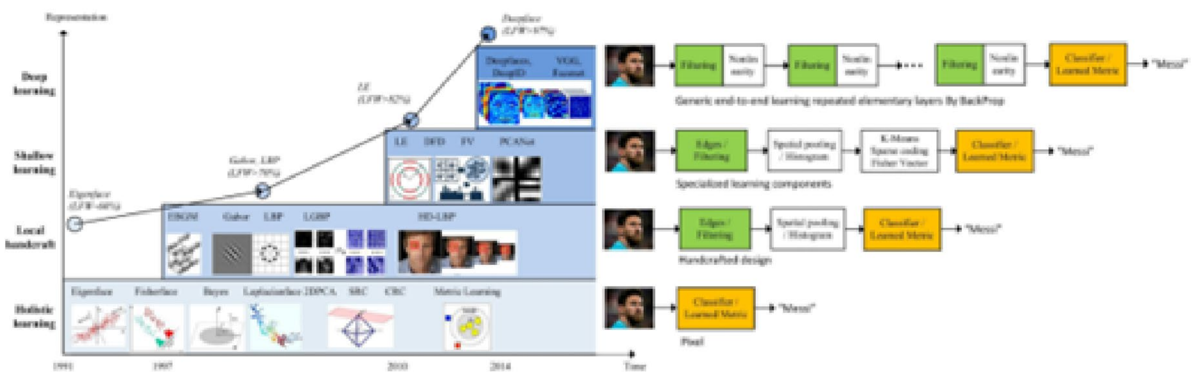


Fig. 7

MILESTONES OF FACE REPRESENTATION FOR RECOGNITION. THE HOLISTIC APPROACHES DOMINATED THE FACE RECOGNITION COMMUNITY IN THE 1990S. IN THE EARLY 2000S, HANDCRAFTED LOCAL DESCRIPTORS BECAME POPULAR, AND THE LOCAL FEATURE LEARNING APPROACHES WERE INTRODUCED IN THE LATE 2000S. IN 2014, DEEPFACE [57] AND DEEPID [58] ACHIEVED A BREAKTHROUGH ON STATE-OF-THE-ART (SOTA) PERFORMANCE, AND RESEARCH FOCUS HAS SHIFTED TO DEEP-LEARNING-BASED APPROACHES. AS THE REPRESENTATION PIPELINE BECOMES DEEPER AND DEEPER, THE LFW (LABELED FACE IN-THE-WILD) PERFORMANCE STEADILY IMPROVES FROM AROUND 60% TO ABOVE 90%, WHILE DEEP LEARNING BOOSTS THE PERFORMANCE TO 99.80%.

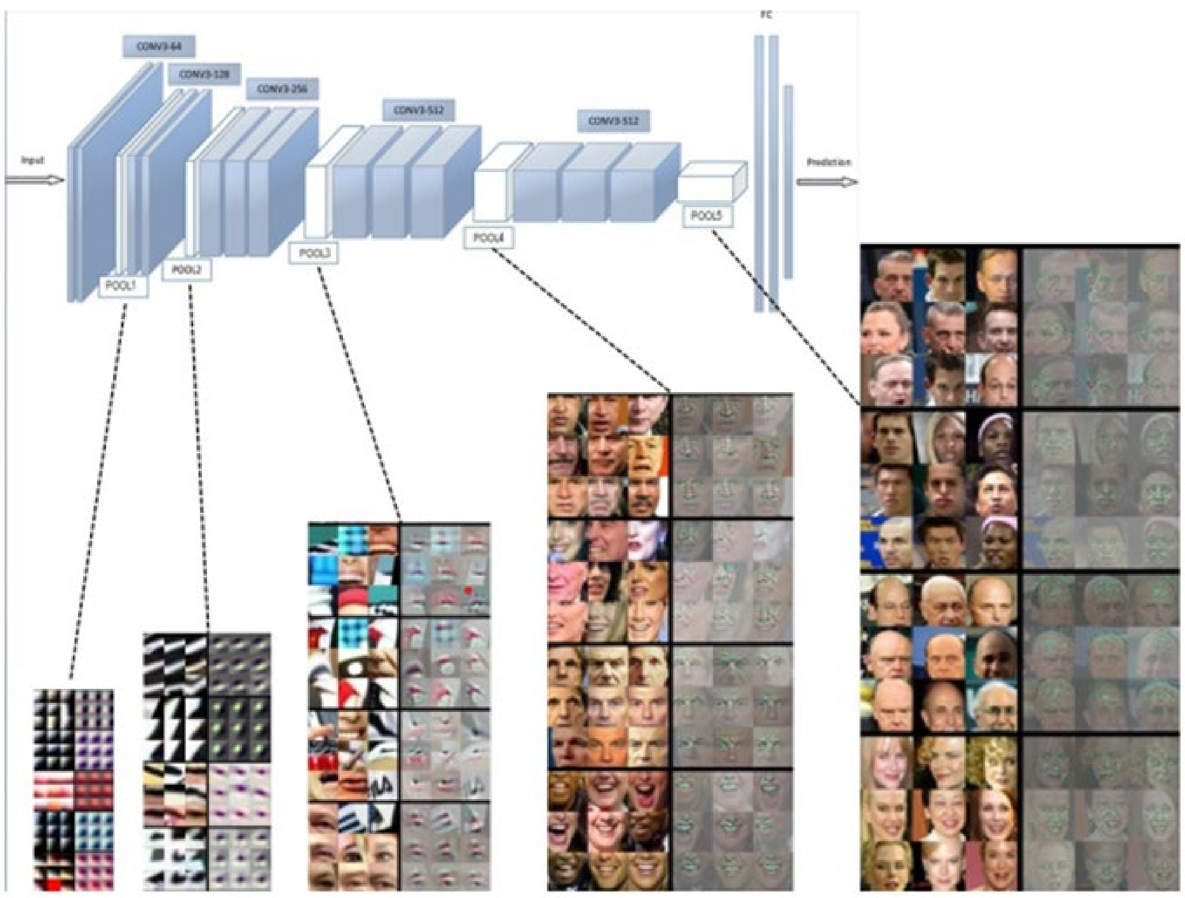


Fig. 8

Overview

A. Components of Face Recognition: there are three modules needed for FR system, as shown in Fig. 9. First, a face detector is used to localize faces in images or videos. Second, with the facial landmark detector, the faces are aligned to normalized canonical coordinates. Third, the FR module is implemented

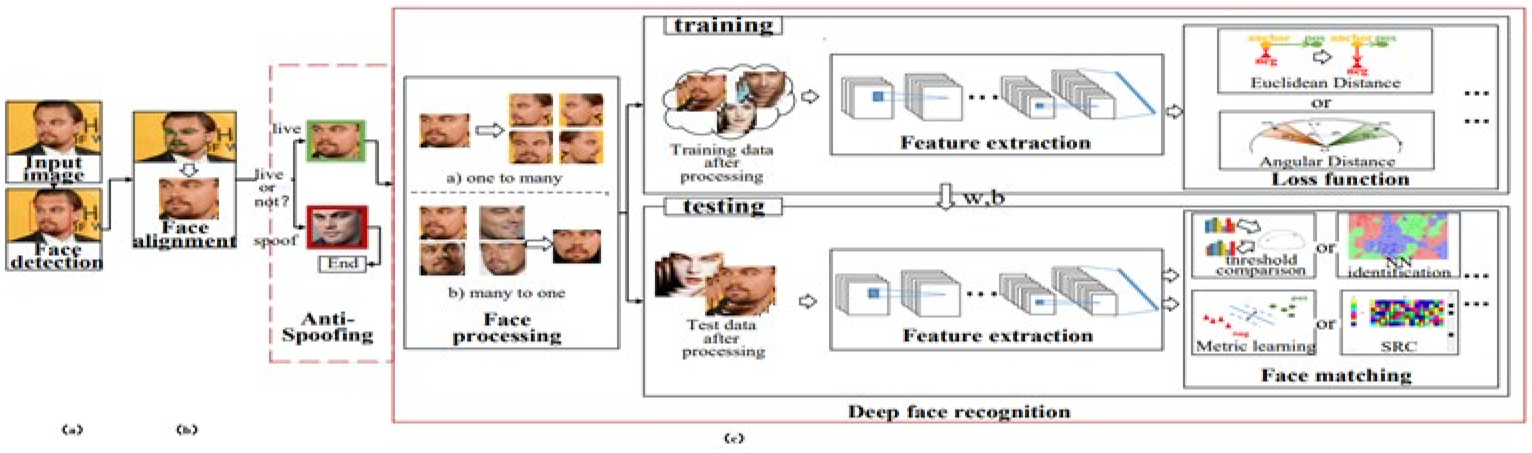


Fig.9

1) Face Processing: Although deep-learning-based approaches have been widely used, various conditions, such as poses, illuminations, expressions and occlusions, still affect the performance of deep FR. Accordingly, face processing is introduced to address this problem. The face processing methods are categorized as "one to-many augmentation" and "many-to-one normalization", as shown in Table I.

· "One-to-many augmentation". These methods generate many patches or images of the pose variability from a single image to enable deep networks to learn pose invariant representations.

· "Many-to-one normalization". These methods recover the canonical view of face images from one or many images of a Non frontal view; then, FR can be performed as if it were under controlled conditions.

TABLE I

DIFFERENT DATA PREPROCESSING APPROACHES

Data processing	Brief Description
one to many	These methods generate many patches or images of the pose variability from a single image
many to one	These methods recover the canonical view of face images from one or many images of nonfrontal view

Data processing Brief Description

one to many These methods generate many patches or images of the pose variability from a single image

many to one These methods recover the canonical view of face images from one or many images of nonfrontal view

2) Deep Feature Extraction: Network Architecture. The architectures can be categorized as backbone and assembled networks, as shown in Table II. Inspired by the extraordinary success on the ImageNet challenge, the typical CNN architectures, e.g., AlexNet, VGGNet, GoogleNet, ResNet and SENet [59], [62], [63], [64], [65], are introduced and widely used as the baseline models in FR (directly or slightly modified). In addition to the mainstream, some assembled networks, e.g., multi-task networks and multi-input networks, are utilized in FR. Hu et al. [66] shows that accumulating the results of assembled networks provides an increase in performance compared with an individual network.

Loss Function. The softmax loss is commonly used as the supervision signal in object recognition, and it encourages the separability of features. However, the softmax loss is not sufficiently effective for FR because intra-variations could be larger than inter-differences and more

discriminative features are required when recognizing different people. Many works focus on creating novel loss functions to make features not only more separable but also discriminative.

3) Face Matching by Deep Features: FR can be categorized as face verification and face identification. In either scenario, a set of known subjects is initially enrolled in the system (the gallery), and during testing, a new subject (the probe) is presented. After the deep networks are trained on massive data with the supervision of an appropriate loss function, each of the test images is passed through the networks to obtain a deep feature representation. Using cosine distance or L2 distance, face verification computes one-to-one similarity between the gallery and probe to determine whether the two images are of the same subject, whereas face identification computes one-to many similarity to determine the specific identity of a probe face. In addition to these, other methods are introduced to postprocess the deep features such that the face matching is performed efficiently and accurately, such as metric learning, sparse-representation-based classifier (SRC), and so forth. To sum up, we present FR modules and their commonly used methods in Fig. 10 to help readers to get a view of the whole FR. In deep FR, various training and testing face databases are constructed, and different architectures and losses of deep FR always follow those of deep object

In deep FR, various training and testing face databases are constructed, and different architectures and losses of deep FR always follow those of deep object classification and are modified according to unique characteristics of FR. Moreover, in order to address unconstrained facial changes, face processing methods are further designed to handle poses, expressions and occlusions variations. Benefiting from these strategies, deep FR system significantly improves the SOTA and surpasses human performance. When the applications of FR become more and more mature in general scenario, recently, different solutions are driven for more difficult specific scenarios, such as cross-pose FR, cross-age FR, video FR.

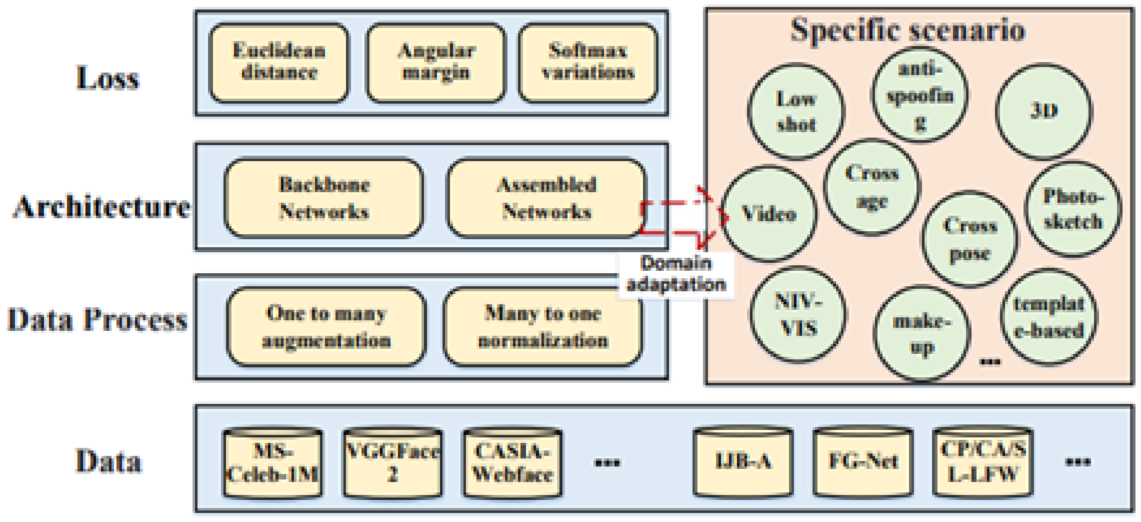


Fig. 10

FR STUDIES HAVE BEGUN WITH GENERAL SCENARIO, THEN GRADUALLY GET CLOSE TO MORE REALISTIC APPLICATIONS AND DRIVE DIFFERENT SOLUTIONS FOR SPECIFIC SCENARIOS, SUCH AS CROSS-POSE FR, CROSS-AGE FR, VIDEO FR. IN SPECIFIC SCENARIOS, TARGETED TRAINING AND TESTING DATABASE ARE CONSTRUCTED, AND FACE PROCESSING, ARCHITECTURES AND LOSS FUNCTIONS ARE MODIFIED BASED ON THE SPECIAL REQUIREMENTS.

For most applications, it is difficult to include the candidate faces during the training stage, which makes FR become a “zero-shot” learning task. Fortunately, since all human faces share a similar shape and texture, the representation learned from a small proportion of faces can generalize well to the rest. Based on this theory, a straightforward way to improve generalized performance is to include as many IDs as possible in the training set. For example, Internet giants such as Facebook and Google have reported their deep FR system trained by 106 − 107 IDs [61], [57].

4) FaceNet: A Unified Embedding for Face Recognition

Unified system for face verification (is this the same person), recognition (who is this person) and clustering (find common people among these faces). This method is based on learning a Euclidean embedding per image using a deep convolutional network. The network is trained such that the squared L2 distances in the embedding space directly correspond to face similarity:

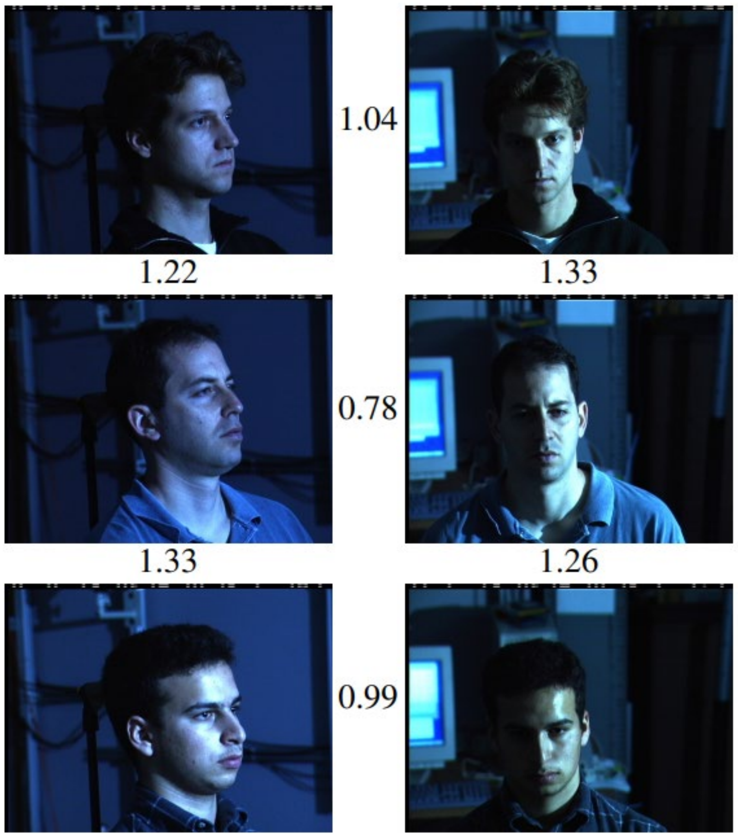


Figure 11. Illumination and Pose invariance. Pose and illumination have been a long-standing problem in face recognition. This figure shows the output distances of FaceNet between pairs of faces of the same and a different person in different pose and illumination combinations. A distance of 0.0 means the faces are identical, 4.0 corresponds to the opposite spectrum, two different identities faces of the same person have small distances and faces of distinct people have large distances.

Once this embedding has been produced, then the aforementioned tasks become straight-forward: face verification simply involves thresholding the distance between the two embeddings; recognition becomes a k-NN classification problem; and clustering can be achieved using off-the shelf techniques such as k-means or agglomerative clustering

Previous face recognition approaches based on deep networks use a classification layer [52, 54] trained over a set of known face identities and then take an intermediate bottle neck layer as a representation used to generalize recognition beyond the set of identities used in training. The downsides of this approach are its indirectness and its inefficiency: one has to hope that the bottleneck representation generalizes well to new faces; and by using a bottleneck layer the representation size per face is usually very large (1000s of dimensions). Some recent work has reduced this dimensionality using PCA, but this is a linear transformation that can be easily learnt in one layer of the network.

In contrast to these approaches, FaceNet directly trains its output to be a compact 128-D embedding using a triplet-based loss function based on large margin nearest neighbour (LMNN).

Triplets consist of two matching face thumbnails and a non-matching face thumbnail and the loss aims to separate the positive pair from the negative by a distance margin. The thumbnails are tight crops of the face area, no 2D or 3D alignment, other than scale and translation is performed.

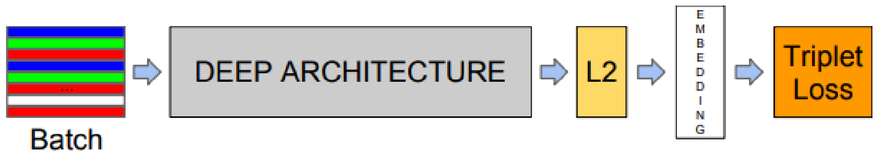


Figure 12 Model structure consists of a batch input layer and a deep CNN followed by L2 normalization, which results in the face embedding. This is followed by the triplet loss during training.

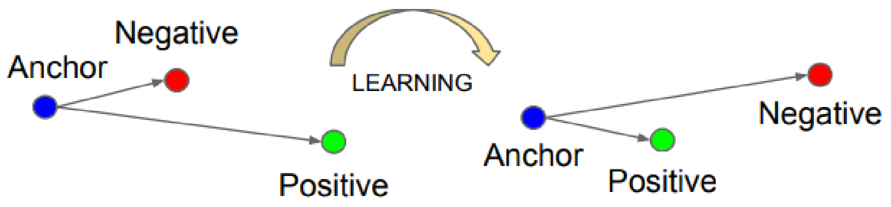


Figure 13. The Triplet Loss minimizes the distance between an anchor and a positive, both of which have the same identity, and maximizes the distance between the anchor and a negative of a different identity

6.2. Networking with Flask and SocketIO

What is Flask?

Flask is a micro framework offering all the basic features of a web application. It's easy to get started on and expand with Python's wealth of libraries and extensions.

On its own, the framework's capabilities are limited and extra functionalities need to be added with supplemental technologies. In many ways, it takes an opposite approach to that of Django, as it enables developers to change and tailor it to their particular needs at will.

With Flask, diversifying the structure of your software project with micro frameworks is effortless. Flexibility is the core feature of this open-source framework.

What is Flask used for?

Well, Flask may be the right choice when you know there will be more technological decisions to make on the way, and you don't want to be limited by the decisions you would've already made.

If that sounds familiar, you're probably considering a whole bunch of solutions, trying to find the best one and go with it. That's perfectly fine. Flask gives you a wide variety of options when you have little to no expectations.

In short, Flask is a great fit in three main instances:

1. Experimentation with the architecture, libraries, and newest technologies.

2. Unknown number of highly specialized (micro) services.

3. Multitude of small, disparate features.

Conversely, Flask also works really well with web apps we know for a fact will be divided into many smaller services that won't require that much code to accomplish their goals. Rather, they have a simple goal supported by simple implementation. An example here would be web applications that fall under larger systems used in DevOps, like monitoring or large-scale logging solutions.

Why use Flask?

Django is opinionated, monolithic, and fixed, making customization more difficult. For instance, using a different ORM framework in Django is possible, but it involves more non-standard work from your developers.

When you're not sure which libraries or frameworks will work best for your software product, it is recommended to use Flask.

Virtually every Flask project out there has its own unique "internal tech stack" of libraries, frameworks, etc. It's something of a free-for-all.

Because of this, Flask is widely known to facilitate experimentation to a high degree. Therefore, in conclusion the Advantages of Flask-based systems are:

- Higher flexibility.
- Higher compatibility with latest technologies.
- High scalability for simple web applications.
- Technical experimentation.
- Customization.
- Slightly higher framework performance.
- Easier to use for simple cases.
- Smaller size of the code base.

We found two options for the server that can be used:

1- Rest API.

2- Socket Server.

Let's discuss both in detail:

6.2.1 Rest API

- Rest API is good for sending requests and receiving responses. But it is one way. The user must send a request to receive a response. The server can't send data to the user directly.

6.2.2 Socket

- Socket is the most popular server that is used in real time applications.
- Implementing a socket server has some difficulties, so we looked for a simple package that would make it easier to implement. That package is called Socket.IO

Socket.IO

Performance: In most cases, the connection will be established with Web Socket, providing a low-overhead communication channel between the server and the client.

Reliable: In case the Web Socket connection is not possible, it will fall back to HTTP long-polling. And if the connection is lost, the client will automatically try to reconnect.

Scalable: Scale to multiple servers and send events to all connected clients with ease.

6.3. Client-side platform

At first, we were planning to use Flutter for our client-side part. But due to package incompatibility between dart-socketio-client and python-socketio we were unable to establish a proper connection to the server. As such we used python for the client-side as it was most familiar and had all the necessary tools.

6.4. Verification Process

A lot of **tools** are used for the verification process which are:

1. OpenCV DNN Single shot detector for face detection as it had the best performance and accuracy balance.

(res10_300x300_ssd_iter_140000.caffemodel)

2. OpenCV Haar cascade classifier for side profile and smile detection.

(haarcascade_profileface.xml, haarcascade_smile.xml)

3. Pretrained FaceNet is used server side to encode faces in the database.

The process is divided into two parts (i) client-side, (ii) server-side.

4. Of our liveness models we decided to use the frame-based ones for real-time performance.

Client-side:

- The client side continually checks for face presence using the OpenCV SSD pretrained model. If face is not found the client restarts operation (to prevent any verification sneak-around)
- The client side continually compares consecutive frames using KL Divergence to check for any rapid movement of abnormalities in the video (make sure KLD (frame_t, frame_t-1) < 0.3).

- While preforming all these checks to ensure continuous face presence in frame the system prompts the user to [smile, look left, look right] in random order. The user has to restart from the beginning if he fails a prompt and prompts are reshuffled (as a measure against replay video attacks).
- If the user fulfills all the prompts the client-side then starts emitting frames to the server for further liveness checking.

Server-side:

- the server receives frames from the client and prepares each frame to feed to the liveness model.
- If the liveness model labels the person as real then the server checks the database of registered faces.
- The face of the person is encoded using FaceNet and compared with every face in the database using cosine distance.
- If the face is within 0.07 of another face in the database, then he is identified as that person and he is logged into the database.
- If the face is not close to any face in the data base the person is rejected.

Conclusion

Facial recognition is a topic that divides opinion. If you were to believe everything you read in the media, you would think that facial recognition technology is universally mistrusted and disliked. This couldn't be further from the truth.

Like most technology, facial recognition has its detractors. Most of the dislike and mistrust comes from a fear that facial recognition technology is an infringement on our personal privacy. Fueled by misinformation in the media, many have jumped on the bandwagon, fearing that once your face has been recognized and identified by facial recognition technology, that hackers can simply break into a database, steal your identity and you can never get it back again. Which is simply not true.

Facial recognition and other biometric technologies are some of the safest ways to authenticate that you are who you say you are. Much safer than a traditional password and much less likely to be stolen. You also can't lose your biometric indicators – your face, your fingerprints and your irises are always with you, making them more practical than passwords for many people.

The truth is, that more and more industries are investing in facial recognition technology as a way of improving security, creating a better user experience and reducing costs. From airlines to automobiles, law enforcement to border control, the practical applications for facial recognition technology, coupled with the enhanced security that biometric authentication offers means that we are all going to become more familiar with facial recognition in future.

The biggest challenge of face recognition is presentation attack which requires well-tuned models. Also, Dataset that satisfies the variation of human faces.

References

[1] Goode Intelligence, I. R. (2021, 4 19). Retrieved from https://www.businesswire.com/news/home/20210419005112/en/Goode-Intelligence-and-ID-RD-Announce-2021-Biometric-Survey-Results

[2] Peter. (2018, 3 12). *Spiceworks*. Retrieved from https://community.spiceworks.com/security/articles/2952-data-snapshot-biometrics-in-the-workplace-commonplace-but-are-they-secure

[3] https://research.fb.com/wp-content/uploads/2016/11/deepface-closing-the-gap-to-human-level-performance-in-face-verification.pdf

[4] https://arxiv.org/pdf/1503.03832.pdf

[5] https://www.youtube.com/watch?v=b9t17Sw56is

[6] W. Bao, H. Li, N. Li, and W. Jiang. A liveness detection method for face recognition based on optical flow field. In Image Analysis and Signal Processing, 2009. IASP 2009. International Conference on, pages 233–236. IEEE, 2009.

[7] J. Li, Y. Wang, T. Tan, and A. K. Jain. Live face detection based on the analysis of Fourier spectra. In In Biometric Technology for Human Identification, pages 296–303, 2004.

[8] M. M. Chakka, A. Anjos, S. Marcel, R. Tronci, D. Muntoni, G. Fadda, M. Pili, N. Sirena, G. Murgia, M. Ristori, et al. Competition on counter measures to 2-d facial spoofing attacks. In Biometrics (IJCB), 2011 International Joint Conference on, pages 1–6. IEEE, 2011.

[9] A. Anjos and S. Marcel. Counter-measures to photo attacks in face recognition: a public database and a baseline. In Biometrics (IJCB), 2011 international joint conference on, pages 1–7. IEEE, 2011.

[10] I. Chingovska, A. Anjos, and S. Marcel. On the effectiveness of local binary patterns in face anti-spoofing. Idiap-RR Idiap-RR-19-2012, Idiap, 7 2012.

[11] Z. Zhang, J. Yan, S. Liu, Z. Lei, D. Yi, and S. Z. Li. A face antispoofing database with diverse attacks. In Biometrics (ICB), 2012 5th IAPR international conference on, pages 26–31. IEEE, 2012.

[12] X. Tan, Y. Li, J. Liu, and L. Jiang. Face liveness detection from a single image with sparse low rank bilinear discriminative model. In Computer Vision–ECCV 2010, pages 504– 517. Springer, 2010.

[13] B. Peixoto, C. Michelassi, and A. Rocha. Face liveness detection under bad illumination conditions. In Image Processing (ICIP), 2011 18th IEEE International Conference on, pages 3557–3560. IEEE, 2011.

[14] Di Wen, Hu Han and Jain, A., 2015. Face Spoof Detection with Image Distortion Analysis. *IEEE Transactions on Information Forensics and Security*, 10(4), pp.746-761.

[15] I. Chingovska, J. Yang, Z. Lei, D. Yi, S. Z. Li, O. Kahm, C. Glaser, N. Damer, A. Kuijper, A. Nouak, et al. The 2nd competition on counter measures to 2d face spoofing attacks. In Biometrics (ICB), 2013 International Conference on, pages 1–6. IEEE, 2013.

[16] J. Maatta, A. Hadid, and M. Pietikainen. Face spoofing detection from single images using micro-texture analysis. In Biometrics (IJCB), 2011 international joint conference on, pages 1–7. IEEE, 2011.

[17] K. Patel, H. Han, A. K. Jain, and G. Ott. Live face video vs. spoof face video: Use of moire patterns to detect replay video attacks. In ICB, pages 98–105. IEEE, 2015.

[18] T. de Freitas Pereira, A. Anjos, J. M. De Martino, and S. Marcel. Lbp-top based countermeasure against face spoofing attacks. In Computer Vision-ACCV 2012 Workshops, pages 121–132. Springer, 2012.

[19] G. Pan, L. Sun, and Z. Wu. Eyeblink-based anti-spoofing in face recognition from a generic webcamera.

[20] W. Kim, S. Suh, and J.-J. Han. Face liveness detection from a single image via diffusion speed model. IEEE Transactions on Image Processing, 4(8):2456–2465, 2015.

[21] H.-Y. Wu, M. Rubinstein, E. Shih, J. Guttag, F. Durand, and W. Freeman. Eulerian video magnification for revealing subtle changes in the world. ACM Trans. Graph., 31(4):65:1– 65:8, July 2012.

[22] R. Tronci, D. Muntoni, G. Fadda, M. Pili, N. Sirena, G. Murgia, M. Ristori, S. Ricerche, and F. Roli. Fusion of multiple clues for photo-attack detection in face recognition systems. In 2011 IEEE International Joint Conference on Biometrics, IJCB 2011, Washington, DC, USA, October 11-13, 2011, pages 1–6, 2011.

[23] A. Agarwal, R. Singh, and M. Vatsa. Face anti-spoofing using haralick features. In Biometrics Theory, Applications and Systems (BTAS), 2016 IEEE 8th International Conference on, pages 1–6. IEEE, 2016.

[24] I. Chingovska, J. Yang, Z. Lei, D. Yi, S. Z. Li, O. Kahm, C. Glaser, N. Damer, A. Kuijper, A. Nouak, et al. The 2nd competition on counter measures to 2d face spoofing attacks. In Biometrics (ICB), 2013 International Conference on, pages 1–6. IEEE, 2013.

[26] J. Yang, Z. Lei, and S. Z. Li. Learn convolutional neural network for face anti-spoofing. CoRR, abs/1408.5601, 2014.

[27] Rehman, Y., Po, L. and Liu, M., 2020. SLNet: Stereo face liveness detection via dynamic disparity-maps and convolutional neural network. *Expert Systems with Applications*, 142, p.113002.

[28] Lakshminarayana, N.N., Narayan, N., Napp, N., Setlur, S. and Govindaraju, V., 2017, February. A discriminative spatio-temporal mapping of face for liveness detection. In *2017 IEEE International Conference on Identity, Security and Behavior Analysis (ISBA)* (pp. 1-7). IEEE.

[29] Lin, B., Li, X., Yu, Z. and Zhao, G., 2019, May. Face liveness detection by rppg features and contextual patch-based cnn. In *Proceedings of the 2019 3rd international conference on biometric engineering and applications* (pp. 61-68).

[30] Y. Moon and I. Ryoo and S. Kim "Face Antispoofing Method Using Color Texture Segmentation on FPGA,"Hindawi Security and Communication Networks, Vol, 2021, Article ID 9939232, 11 pages,2021. https://doi.org/10.1155/2021/9939232.

[31] Z. Boulkenafet, J. Komulainen, and A. Hadid, "Face antispoofing based on color texture analysis," in Proceedings of the 2015 IEEE International Conference on Image Processing (ICIP), pp. 2636–2640, Quebec, Canada, September 2015.

[32] S. H. Lee, H. Kim, and Y. M. Ro, "A comparative study of color texture features for face analysis," in Computational Color Imaging. CCIW, S. Tominaga, R. Schettini, and A. Tremeau, ´ Eds., Berlin, Heidelberg, Springer.

[33] G. Kim, S. Eum, J. Suhr, D. Kim, K. Park, and J. Kim, "Face liveness detection based on texture and frequency analyses, in Proceedings of 2012 5th IAPR International Conference on Biometrics, ICB, pp. 67–72, New Delhi, India, April 2012.

[34] G. Sang, L. Jing, and Q. Zhao, "Pose-invariant face recognition via RGB-D images," Computational Intelligence and

[35] T. Ojala, M. Pietikainen, and D. Harwood, "Performance evaluation of texture measures with classification based on Kullback discrimination of distributions," in Proceedings of the 12th IAPR International Conference on

Pattern Recognition Conference A: Computer Vision & Image Processing,, pp. 582–585, Jerusalem, Israel, October 1994.

[36] J. Galbally, S. Marcel, and J. Fierrez, “Biometric antispoofing methods: a survey in face recognition,” IEEE Access, vol. 2, pp. 1530–1552, 2014.

[37] A. Krizhevsky, I. Sutskever, and G. Hinton, “ImageNet classification with deep convolutional neural networks,” Neural Information Processing Systems, vol. 25, 2012.

[38] M. Turk and A. Pentland, “Eigenfaces for recognition,” Journal of cognitive neuroscience, vol. 3, no. 1, pp. 71–86, 1991.

[39] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, “Eigenfaces vs. fisherfaces: Recognition using class specific linear projection,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 19, no. 7, pp. 711–720, 1997.

[40] B. Moghaddam, W. Wahid, and A. Pentland, “Beyond eigenfaces: probabilistic matching for face recognition,” Automatic Face and Gesture Recognition, 1998. Proc. Third IEEE Int. Conf., pp. 30–35, Apr 1998.

[41] W. Deng, J. Hu, J. Lu, and J. Guo, “Transform-invariant pca: A unified approach to fully automatic facealignment, representation, and recognition,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 36, no. 6, pp. 1275–1284, June 2014.

[42] X. He, S. Yan, Y. Hu, P. Niyogi, and H.-J. Zhang, “Face recognition using laplacianfaces,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 27, no. 3, pp. 328–340, 2005.

[43] S. Yan, D. Xu, B. Zhang, and H.-J. Zhang, “Graph embedding: A general framework for dimensionality reduction,” Computer Vision and Pattern Recognition, IEEE Computer Society Conference on, vol. 2, pp. 830–837, 2005.

[44] W. Deng, J. Hu, J. Guo, H. Zhang, and C. Zhang, “Comments on “globally maximizing, locally minimizing: Unsupervised discriminant projection with applications to face and palm biometrics”,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 30, no. 8, pp. 1503–1504, 2008.

[45] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma, “Robust Face Recognition via Sparse Representation,” IEEE Trans. Pattern Anal. Machine Intell., vol. 31, no. 2, pp. 210–227, 2009.

[46] L. Zhang, M. Yang, and X. Feng, “Sparse representation or collaborative representation: Which helps face recognition?” in 2011 International conference on computer vision. IEEE, 2011, pp. 471−478.

[47] W. Deng, J. Hu, and J. Guo, "Extended src: Undersampled face recognition via intraclass variant dictionary," IEEE Trans. Pattern Anal. Machine Intell., vol. 34, no. 9, pp. 1864–1870, 2012.

[48] ——, "Face recognition via collaborative representation: Its discriminant nature and superposed representation," IEEE Trans. Pattern Anal. Mach. Intell., vol. PP, no. 99, pp. 1–1, 2018.

[49] C. Liu and H. Wechsler, "Gabor feature based classification using the enhanced fisher linear discriminant model for face recognition," Image processing, IEEE Transactions on, vol. 11, no. 4, pp. 467–476, 2002.

[50] T. Ahonen, A. Hadid, and M. Pietikainen, "Face description with local binary patterns: Application to face recognition," IEEE Trans. Pattern Anal. Machine Intell., vol. 28, no. 12, pp. 2037–2041, 2006.

[51] W. Zhang, S. Shan, W. Gao, X. Chen, and H. Zhang, "Local gabor binary pattern histogram sequence (lgbphs): A novel non-statistical model for face representation and recognition," in ICCV, vol. 1. IEEE, 2005, pp. 786–791.

[52] D. Chen, X. Cao, F. Wen, and J. Sun, "Blessing of dimensionality: High-dimensional feature and its efficient compression for face verification," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2013, pp. 3025–3032.

[53] W. Deng, J. Hu, and J. Guo, "Compressive binary patterns: Designing a robust binary face descriptor with random-field eigenfilters," IEEE Trans. Pattern Anal. Mach. Intell., vol. PP, no. 99, pp. 1–1, 2018.

[54] Z. Cao, Q. Yin, X. Tang, and J. Sun, "Face recognition with learningbased descriptor," in 2010 IEEE Computer society conference on computer vision and pattern recognition. IEEE, 2010, pp. 2707–2714.

[55] Z. Lei, M. Pietikainen, and S. Z. Li, "Learning discriminant face descriptor," IEEE Trans. Pattern Anal. Machine Intell., vol. 36, no. 2, pp. 289–302, 2014.

[56] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma, "Pcanet: A simple deep learning baseline for image classification?" IEEE Transactions on Image Processing, vol. 24, no. 12, pp. 5017–5032, 2015

[57] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 1701–1708.

[58] Y. Sun, Y. Chen, X. Wang, and X. Tang, “Deep learning face representation by joint identification-verification,” in Advances in neural information processing systems, 2014, pp. 1988–1996. [22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Advances in neural information processing systems, 2012, pp. 1097–1105.

[60] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, “Labeled faces in the wild: A database for studying face recognition in unconstrained environments,” Technical Report 07-49, University of Massachusetts, Amherst, Tech. Rep., 2007.

[61] F. Schroff, D. Kalenichenko, and J. Philbin, “Facenet: A unified embedding for face recognition and clustering,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 815–823.

[62] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” arXiv preprint arXiv:1409.1556, 2014.

[63] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1–9.

[64] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778

[65] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 7132–7141.

[66] G. Hu, Y. Yang, D. Yi, J. Kittler, W. Christmas, S. Z. Li, and T. Hospedales, “When face recognition meets with deep learning: an evaluation of convolutional neural networks for face recognition,” in ICCV workshops, 2015, pp. 142–150.