Spoofing Attack Detection with Convolutional Neural Networks

Andrea Maranesi,[1] *Alessandro* Muscatello[1]

[1] *Department of Information Engineering, Università Politecnica delle Marche*

Abstract. Face Recognition system are widely used since they are extremely convenient for the end user and compared with other system linke fingerprint, iris or voice authentication that are more introusive. Unfortunately FR systems are known for suffering of Spoofing Attack that are threat to be considered. This paper aims to create a CNN Network that is capable of be trained on various spoofing attack databases.

1 Introduction

Since there is no need of privileged access to the interact with the authentication systems, they are considerable to be the most susceptible to attacks. We can call attacks on Biometric System as Presentation Attacks. The ISO definition of PA is "a presentation to the biometric data capture subsystem with the goal of interfering with the operation of the biometric system". This means that both the impersonation and obfuscation are PAs. In fact, some attackers just could want to avoid to be correctly identified. There are many dataset publicly available to use as source for training algorithms that are capable of detect those attacks. We will create a dataset for our testing in conjunction with other dataset. Starting form an existing architecture and using some tecniques used on widely used networks we will create a new CNN architecture that will be capable of generalize on different spoofing attack dataset. We will show our testing results on different dataset and different architecture and try to explain why certain parameters influence the reuslts.

2 Related Work

Until today, different approaches have been tried to get good results on the anti-spoofing task, using different datasets as training data. Many of them use hand-crafted features, instead of pure CNNs, and they are described in [1]. However, they require a lot of data pre-processing and there are also other issues as reported in [2]. The main reason of using CNNs is the increasing number of data inside datasets, and the complexity and variability of the tasks.

2.1 CNN Approaches

In [1] authors use an approach in which they detect faces from original datasets in different scales, using OpenCV as tool. This was done to contain some background data. They used a common CNN used for ImageNet classification, and they trained from scratch. The output of the first and second convolutional layers are normalized. To avoid over-fitting, the first two fully-connected layers are followed by two dropout layers, and the output has softmax as activation function. They firstly train the CNN, then they fed the results of the last fully-connected layer to train an SVM classifier.

In [2], instead, they try to use Inception-v3, ResNet50 and ResNet152 for face anti-spoofing. They train from scratch and also using transfer learning. ResNet512, based on their results, trained from scratch, using a low learning rate, seems to reach good results on MFSD dataset.

As regards [3], the author uses his own dataset, created by recording his face, and his phone in which he shows his face. The initial approach is the same of the previous two works: extracting faces from images, then feed them, without normalization in this case, into a shallow network, made by two convolutional stages, followed by two fully-connected layers. The output has softmax as activation function.

3 Materials

We extracted faces from each dataset using an SSD Network. (figure 9)

3.1 Our Dataset

3.1.1 Description

We took videos mainly from YouTube, that show real people in front of the camera. We then acquire the same video using a mobile phone, while the same frames were being played on a laptop or another mobile phone.

3.1.2 Dataset Structure

It consists of 84 videos organized in the following way:

1. 40 for training: 15 for bona fide, 16 for attackers.
2. 25 for testing: 25 for bona fide, 28 for attackers.

We extracted frames from each image, balancing both training and testing with undersampling.

Dataset Dimensions	4656
Class distribution	50/50
Training/Validation split	48/52
Training images	2238
Validation images	2418
Training class distribution	50/50
Validation class distribution	50/50

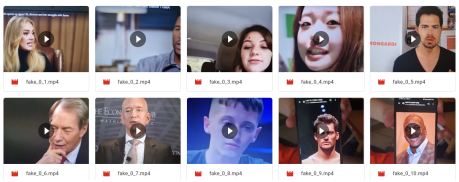


Figure 1: Example of our videos

3.2 CSMAD

3.2.1 Description

Face-biometric data has been collected from 14 subjects to create this dataset. Subjects participating in this data-collection have played three roles: targets, attackers, and bona-fide clients. The subjects represented in the dataset are referred to here with letter-codes:A .. N. The subjects A..F have also been targets. That is, face-data for these six subjects has been used to construct their corresponding flexible masks (made of silicone). These masks have been made by Nimba Creations Ltd., a special effects company [4].

3.2.2 Dataset Structure

We got a subset of the original dataset, taking different subjects and increasing the variability of data. This means we didn't get a lot of equal

frames of the same person.This was done to try to train faster and with less pre-processing.

Training and validation data is balanced using undersampling. We used for training and testing the following subjects:

A,C as attacking subjects (stand/wear) for training; B,D for testing (stand/wear)

A,C,H,E,F,G as bona fide subjects; B,D,I for testing.

Dataset Dimensions	2448
Class distribution	50/50
Training/Validation split	66/34
Training images	1632
Validation images	816
Training class distribution	50/50
Validation class distribution	50/50

Figure 2: Example of CSMAD data

3.3 3DMAD

3.3.1 Description

The 3D Mask Attack Database (3DMAD) is a biometric (face) spoofing database. We get a subset of data. In our case it contains frames of 17 persons, recorded using Kinect for both real-access and spoofing attacks [5]. Each frame consists of:

a depth image (640x480 pixels – 1x11 bits)

the corresponding RGB image (640x480 pixels – 3x8 bits)

The data is collected in 3 different sessions for all subjects and for each session 5 videos of 300 frames are captured. The recordings are done under controlled conditions, with frontal-view and neutral expression. The first two sessions are dedicated to the real access samples, in which subjects are recorded with a time delay of 2 weeks between the acquisitions. In the third session, 3D mask attacks are captured by a single operator (attacker).

3.3.2 Dataset Structure

We got all people for our training, but a subset of the original dataset, excluding too many equal frames.

Training and validation data is balanced using undersampling. We used for testing subjects

7,10,11,12,13,14 while for training the remaining ones.

Dataset Dimensions	2100
Class distribution	50/50
Training/Validation split	63/37
Training images	1320
Validation images	780
Training class distribution	50/50
Validation class distribution	50/50

Figure 3: Example of 3DMAD data

It has been used offline data augmentation to increase the number of the training images.

Dataset Dimensions	4914
Class distribution	50/50
Training/Validation split	67/33
Training images	3276
Validation images	1924
Training class distribution	50/50
Validation class distribution	50/50

Figure 4: Example of MS-Spoof data

3.4 MS-Spoof

3.4.1 Description

Multispectral-Spoof contains face images and printed spoofing attacks recorded in Visible (VIS) and Near-Infrared (NIR) spectra for 21 identities [6].

3.4.2 Dataset Structure

We get images from the original dataset, organizing them in attack and real folders.

We trained and tested using both NIR and VIR acquisitions together.

Training and validation data is balanced using undersampling. We used for training and testing images as suggested by the authors of the dataset.

3.5 Replay-Attack

3.5.1 Description

The Replay-Attack Database for face spoofing consists of 1300 video clips of photo and video attack attempts to 50 clients, under different lighting conditions. This Database was produced at the Idiap Research Institute, in Switzerland [7].

3.5.2 Dataset Structure

We got many frames from almost all the images and videos of the original dataset, so in our case the dataset is organized in two main subdirectories: attack, real.

Training and validation data is balanced using undersampling. We used for training and

testing images as suggested by the authors of the dataset.

Dataset Dimensions	10804
Class distribution	50/50
Training/Validation split	82/18
Training images	8880
Validation images	1924
Training class distribution	50/50
Validation class distribution	50/50

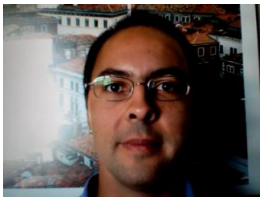


Figure 5: Example of Replay-Attack data

4 AttackNet

We started from CNN used in [3]. We call this architecture **LivenessNet**.

Our aim is to create a novel light architecture to predict bona fide and attacker classes, and this will be done for some reasons:

1. get good results on different datasets.
2. have fast inference time, important for real-time detection.
3. possibility to be implemented on cheap hardware.

4.1 AttackNet v1

Keeping two convolutional steps, for each step we added another convolutional layer and, following the ResNet example, shirt skip connections, with concatenation function (figure 6).

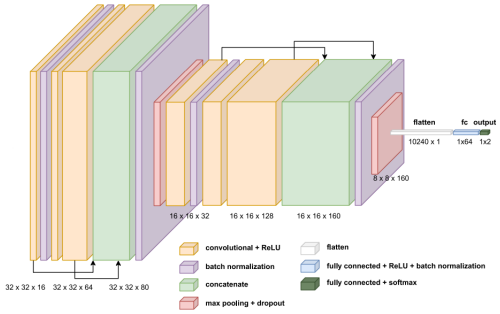


Figure 6: AttackNet v1

4.2 AttackNet v2.1

Keeping skip connections, respect the first one, we introduce LeakyRelu, Hyperbolic Tangent activation function on the first fully-connected layer. Since BatchNormalization can produce negative values in the standardized distribution, with ReLU we can lose information, since it introduces certain sparsity in the network. LeakyReLU tries to solve this issue. Hyperbolic Tangent, furthemore, needs us to output values between -1 and 1, taking any value between -infinite to +infinite. The vanishing gradient problem that tanh can cause is solved using skip connections. (figure 7)).

This network is always used in our dataset, Replay-Attack and MS-Spoof.

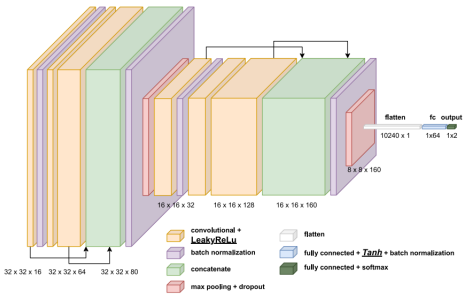


Figure 7: AttackNet v2.1

4.3 AttackNet v2.2

The shirt skip connections uses the add function, instead of concatenation on third axis (figure 8).

This network is only used for CSMAD and 3DMAD.

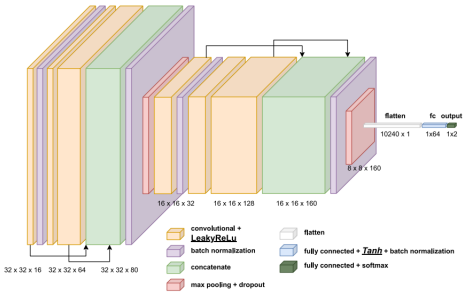


Figure 8: AttackNet v2.2

5 Training

5.1 Images Pre-Processing

All our experiments are made using the library Tensorflow in Python. Each dataset is made of single faces detected from original frames/images using a trained a SSD-ResNet for human face recognition. Each face is then resized, with OpenCV, to 32x32 pixels. For our dataset, Replay-Attack, MS-Spoof we normalized considering the single image: `image/np.max(image`. For CSMAD, 3DMAD we wanted to keep RGB scale and we normalized considering the maximum possible value (255): `image/255`

Figure 9: Resized Face

5.2 Data Augmentation

For data augmentation, we tested using online and offline augmentation. We flipped, rotated, shifted along width and height, zoomed, sheared, horizontal flipped the original images. For each case, we set up slightly different parameters for randomity of these transformations. For each batch, both for online and offline augmentation, we used keras ImageDataGenerator class.

5.3 Optimizers

In this case, we decided to use SGD and Adam optimizers in particular.

6 Results

In the following tables we summarized our best results. We used Colab Pro for training, with an average used Ram for each training session of 4GB, and 1 GB of GPU. With "Acc." we refer to Accuracy; "Prec." = Precision; "FT" = Fine-Tuning; B/A = Bona fide / Attack; BS = Batch Size; depth=using depth as 4° dimension

* = using EarlyStopping

Network	Dataset	Opt.	Mom	LR	Epochs	BS
LivenessNet	Our	SGD	0.5	1e-5	30	16
LivenessNet	CSMAD (RGB)	SGD	0.5	1e-4	20	32
AttackNet v1	Our	SGD	0.5	1e-5	30	32
AttackNet v2.1	Our	Adam	X	1e-6	25	32
AttackNet v2.1	Replay-Att	Adam	X	1e-4	20	64
AttackNet v2.1	Replay-Att (FT)	Adam	X	1e-4	100*	256
AttackNet v2.2	CSMAD (RGB)	Adam	X	1e-5	20	32
AttackNet v2.2	CSMAD (depth)	Adam	X	1e-5	32*	32
AttackNet v2.2	3DMAD (RGB)	Adam	X	1e-5	20	16
AttackNet v2.1	MS-Spoof	Adam	X	1e-4	15	16

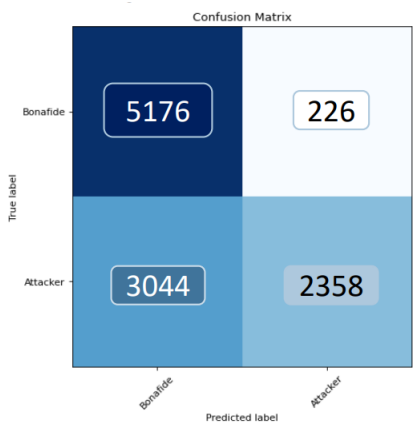
Network	Dataset	Acc.	Prec. B/A		Recall B/A		F1 Score	
LivenessNet	Our	0.76	0.75	0.83	0.86	0.67	0.79	0.74
LivenessNet	Our	0.85	0.77	**1.0**	**1.0**	0.7	0.87	0.82
AttackNet v1	Our	0.80	0.77	0.85	0.87	0.74	0.82	0.79
AttackNet v2.1	Our	**0.87**	**0.8**	**0.89**	**0.9**	**0.77**	**0.85**	**0.83**
AttackNet v2.1	Replay-Att	**0.96**	0.99	0.94	0.93	0.99	0.96	0.96
AttackNet v2.1	Replay-Att (FT)	0.95	**1.0**	**0.95**	**0.94**	**1.0**	**0.97**	**0.97**
AttackNet v2.2	CSMAD (RGB)	0.87	**0.89**	0.85	0.85	**0.89**	0.87	0.87
AttackNet v2.2	CSMAD (depth)	**0.89**	0.88	0.91	0.91	0.87	**0.89**	**0.89**
AttackNet v2.2	3DMAD	1.0	1.0	1.0	1.0	1.0	1.0	1.0
AttackNet v2.1	MS-Spoof	0.95	0.92	0.99	0.99	0.91	0.95	0.95

6.1 Generalization and Fine-Tuning

We would like to focus on Attacknet v2.1 trained on our custom dataset.
The following figure shows a real-use case about AttackNet v2.1 trained on our custom dataset:

Figure 10: Real-use case example

We tested this model also on Replay-Attack training and testing images, obtaining this result



Fine-Tuning Replay-Attack, freezing the first convolutional stage, using Adam, LR of 1e-4, 20 epochs and a batch size of 256.

Acc.	Prec. B/A		Recall B/A		F1 Score	
0.95	0.97	0.93	0.93	0.98	0.95	0.95

7 Conclusions

AttackNet v2.1 and v2.2 seem to work better than previous architectures.
CSMAD was the only case in which depth improved results, but it proved to be a best practice including more information during training. Replay-Attack, CSMAD, MS-Spoof work better using low randomity inside ImageDataGenerator (<=0.2) for data augmentation. In particular, transforming too much the images can generate more noise rather than different useful cases, that is the original purpose. This is a fast architecture, with an inference time of 0.02-0.05ms using low RAM Memory (< 3GB), given a face, so the ability of generalization is limited using only two convolutional steps and two fully-connected layers.

Moreover, the use of Dropout helped to avoid overfitting on paricular features.

8 Future Work

Since we trained from scratch, it could have helped using transfer learning as done in some previous works in the state of the art.
A future work may be improving the architecture, keeping it light, and utilize different datasets than ours during the training phase, given the fact that the datasets shown in this work are old and possibly outdated.
Another interesting point could be to investigate on how the differnt network layers are extracting valuable feature from the images and give meaning to them.

References

[1] J. Yang, Z. Lei, S.Z. Li, *Learn convolutional neural network for face anti-spoofing* (2014), `https://arxiv.org/abs/1408.5601`

[2] C. Nagpal, S.R. Dubey, *A performance evaluation of convolutional neural networks for face anti spoofing* (2018), `https://arxiv.org/abs/1805.04176`

[3] A. Rosebrock, *Live demonstration: Convolutional neural* (2019), `https://pyimagesearch.com/2019/03/11/liveness-detection-with-opencv/`

[4] A.M. Sushil Bhattacharjee, S. Marcel, *"Spoofing Deep Face Recognition With Custom Silicone Masks." in Proceedings of International Conference on Biometrics: Theory, Applications, and Systems (BTAS)* (2018), Vol. 10.1109/BTAS.2018.8698550, http://publications.idiap.ch/index.php/publications/show/3887

[5] N. Erdogmus, S. Marcel, *"Spoofing in 2D Face Recognition with 3D Masks and Anti-spoofing with Kinect", Biometrics: Theory, Applications and Systems* (2013), Vol. 10.1109/BTAS.2013.6712688, https://publications.idiap.ch/index.php/publications/show/2657

[6] A.A.S.M. I. Chingovska, N. Erdogmus, *"Face Recognition Systems Under Spoofing Attacks", in Springer "Face Recognition Across the Imaging Spectrum" (Editor Thirimachos Bourlai)* (2016), https://publications.idiap.ch/index.php/publications/show/3539

[7] S.M. I. Chingovska, A. Anjos, *"On the Effectiveness of Local Binary Patterns in Face Anti-spoofing"; IEEE BIOSIG,* (2012), https://ieeexplore.ieee.org/document/6313548http://publications.idiap.ch/index.php/publications/show/2447