# Deep Learning Based Face Liveliness Detection


Aleksandr Kuznetsov
Department of Political Sciences, Communication and International Relations, University of Macerata, Macerata, Italy, V. N. Karazin Kharkiv National University, Kharkiv, Ukraine
kuznetsov@karazin.ua
https://orcid.org/0000-0003-2331-6326

Andrea Maranesi
Department of Information Engineering, Marche Polytechnic University, Ancona, Italy
S1104219@studenti.univpm.it

Alessandro Muscatello
Department of Information Engineering, Marche Polytechnic University, Ancona, Italy
S1107707@studenti.univpm.it

Davyd Kvaratskheliia
Department of Applied Mathematics, V. N. Karazin Kharkiv National University, Kharkiv, Ukraine
davyd.school@gmail.com

Luca Romeo
Department Economics and Law, University of Macerata, Macerata, Italy
Department of Information Engineering, Marche Polytechnic University, Ancona, Italy
luca.romeo@unimc.it

Riccardo Rosati
Department of Information Engineering, Marche Polytechnic University, Ancona, Italy
r.rosati@pm.univpm.it



***Abstract*—Modern systems of face recognition (FRS) are used in a wide range of computer applications: for user authentication; in email marketing; in social networks; for personal identification and more. However, such technologies are often vulnerable to spoofing attacks. Facial image can be faked in various ways: print a photo; record a video; create a high-quality silicone mask, etc. By presenting a fake to the FRS the attacker has an intention of passing himself off as another person, for instance, trying to get an access to a secure computer system. Face Liveliness Detection solves this problem by detecting whether the person in front of the camera is real or fake. In this article, we explore the possibilities of using deep learning technology for face liveliness detection. We consider several models and setting numerous experiments. As datasets for experiments we use various fake images and for each such dataset we obtained evaluation of the effectiveness. In our research, our main goal is to improve basic deep learning architectures using the latest technologies to get a more accurate model.**



***Keywords—deep learning; face liveliness detection; spoofing attacks; facial recognition system***


## I. Introduction

The Face Recognition System (FRS) uses computer technology of matching the digital image of human faces to accurately identify and measure features, for instance, by checkpoints [1]–[3]. The technology is based, as a rule, on the selection of important features of human faces, their parameters and specific values. Further, based on a comparison of these features with reference copies, a decision is made about whether these features belong to one or another user [4], [5]. In user identification systems, feature sets are used to predict the user's identity [6], [7].

The most promising direction in this area is deep learning methods [8]. In this paper, we are proposing several improvements on some basic deep learning models that allow us to achieve better performance in spoofing detection.

## II. Related Works

A large number of scientific publications are devoted to the study of methods for detecting liveliness using artificial intelligence.

The work [8] provides an overview of convolutional neural networks, sets out in detail the settings and instructions for using deep learning for liveliness detection. The [9] proposes an end-to-end Convolutional Neural Network (CNN) architecture that directly maps raw input face images to the corresponding output classes. The authors tested attacks using distorted printed photographs, printed photographs with eyes cut out, and some video attacks. In [10], new spoofing attacks are proposed based on the transformation of visible face images into a thermal spectrum. At the same time, in comparison with known attacks using silicone masks, the frequency of detection errors increases. In [11], presentational attacks in the near infrared (NIR) image channel were studied using 3D masks made of paper, latex, and silicone. The authors consider complex texture features from the lower layers of a convolutional neural network and show some positive test results. In [12] provides an overview of modern approaches to spoofing attacks using 3D masks, as well as anti-spoofing methods, including existing databases and countermeasures. The article [13] proposes a set of video data of the movement of a face in a silicone mask, containing 200 videos with a real face and 200 videos with a face from a silicone mask. To detect fakes, the authors use a support vector machine (SVM). In [14], the performance of a CNN was evaluated to protect against face spoofing. The authors explored the Inception and ResNet CNN architectures.

Thus, different authors have explored different datasets simulating specific spoof attack scenarios. In addition, many authors have used various spoofing detection efficiency metrics. We propose an advanced deep learning model that performs better on different datasets.

## III. MATERIALS

In this study, we consider various spoofing attack scenarios and related datasets.

### A. *Custom Silicone Mask Attack (CSMAD)*

In this study, we consider various spoofing attack scenarios and related datasets. A.Custom Silicone Mask Attack (CSMAD) The first dataset we consider is CSMAD [15], [16]. The set contains simulated presentation attacks made from six custom-made silicone masks. These are high quality silicone masks made by Nimba Creations Ltd., a special effects company. The dataset was collected at the Idiap Research Institute and is intended for face presentation attack detection experiments. The structure of this data set is shown in Table I.

TABLE I. STRUCTURE OF CSMAD DATASET

| | |
|---|---|
| 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 |

### B. *The 3D Mask Attack Database (3DMAD)*

The second data set used in our work was 3DMAD [17], [18]. It is a database of 3D biometric data (human faces) recorded using Kinect. For shaping the dataset, 17 face masks were taken from ThatsMyFace.com. The structure of this dataset is shown in Table II.

TABLE II. STRUCTURE OF 3DMAD DATASET

| | |
|---|---|
| Dataset dimension | 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 |

### C. *Multispectral-Spoof Database (MSSPOOF)*

Another set we used contains visible (VIS) and near infrared (NIR) face images for 21 personalities. The Multispectral-Spoof set [19], [20] has the structure shown in Table III.

TABLE III. STRUCTURE OF THE MSSPOOF DATASET

| | |
|---|---|
| Dataset dimensions | 4914 |
| Class distribution | 50 / 50 |
| Training / validation split | 67 / 33 |
| Training images | 3276* |
| Validation images | 1638 |
| Training class distribution | 50 / 50 |
| Validation class distribution | 50 / 50 |

* Off-line data augmentation was used to increase the number of training images

### D. *Replay-Attack Database*

The Replay-Attack database [21], [22] consists of 1300 videos 50 with photo and video attack attempts in different numbers of the United States. The Replay-Attack set has the representation shown in Table IV.

TABLE IV. REPLAY-ATTACK DATASET STRUCTURE

| | |
|---|---|
| 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 |

### E. *Our Database*

Our Dataset is made by two sets of images (bona fide and attackers): the first set contains images extracted from videos that shows ‘real’ people. Those videos are taken with a smartphone or downloaded from internet. The second set contains images extracted from videos taken with a laptop webcam that filmed the playback of the first set of videos on the smartphone screen (or vice versa). The structure of our set is shown in Table V.

TABLE V. STRUCTURE OF OUR DATASET

| | |
|---|---|
| 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 |

## IV. METHODS

Reference [8] provides a detailed guide to setting parameters and using deep learning methods for liveliness detection. The author uses the OpenCV computer vision algorithm library, which shows good results in detecting fakes.

Optimization algorithms set rules for the functions according to which the parameters of the neural network change at each epoch. In our study, we use two optimizers: mini-batch SGD and Adam. The work [23] considered the effectiveness of optimizers on different architectures of neural networks and datasets. Both mini-batch and Adam consistently show good results and high execution speed. We use keras.optimizers.SGD to implement mini-batch SGD.

The principle of operation of mini-batch SGD is based on the fact that at each step, only M features are used for destructuring and changing the parameters of the neural network, where M is the batch size (BS). Thus, the mini-batch SGD algorithm approaches the local minimum less smoothly than batch SGD, but in practice approaches the local minimum in fewer iterations. Batch size (BS) is a hyperparameter, usually BS which is a power of two is used, it speeds up model training.

Together with Stochastic Gradient Descent, we use the Momentum method. As already mentioned, SGD approaches the loss function to the local minimum rather jumpy, which worsens the performance. To provide smoothness, we use the Momentum technique, which is based on exponential smoothing. After changing the parameters of the neural network, they change depending on the previous values according to the following formulas:

- Weight change for SGD without Momentum:

$$W = W - \alpha \frac{\partial cost}{\partial W};$$

- Weight change for SGD with Momentum:

for t in iterations:

$$V_{wt} = \beta V_{dW_{t-1}} + (1-\beta) dW;$$

$$w_t = w_{t-1} - \alpha V_{dw_t}.$$

Hyperparameter β – Momentum (Mom.). The closer Momentum is to 1, the smoother the movement to the local minimum.

Another optimization algorithm used in experiments is Adam. Adam combines two methods: Momentum and Root Mean Square Propagation (RMSP). According to the work [24], in most cases it exceeds the efficiency of Momentum and RMSP separately.

Hyperparameter α – learning rate (LR). The larger the LR, the larger the steps the optimization algorithm takes, however, if the LR is too large, it may lead to the fact that the local minimum will not be reached.

### *A. Training and test data extraction*

In every dataset except Our dataset groups of images associated with different subjects were randomly distributed in training and test dataset. Therefore data leakage is not present since two different images from the same group can't be in training and test dataset simultaneously.

### *B. 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.

We are describing the several architectures we used: for each of them, we used Dropout after each Max Pooling layer and after the first hidden layer to avoid to overfit quickly during training.

### *C. Data Augmentation*

For data augmentation, we tested using on-line 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.

### *D. LivenessNet*

We started from CNN used in [8]. We call this architecture LivenessNet (fiure 1). LivenessNet is a conventional convolutional neural network. To avoid overfitting, the number of LivenessNet layers is small, that is, the neural network is shallow.

### *E. AttackNet v1*

To improve the results of [8], the neural network architecture was complicated (figure 2). For every convolutional step, one convolutional layer was added. We were inspired by VGG16 that uses, in the deepest layers, three convolutional stages before pooling. Furthermore, skip connections was used to avoid the problem of "vanishing gradient" and to avoid overfitting: this was done considering the ResNet example.

### *F. AttackNet v2.1*

After the first improvement, we did other changes to improve the efficacy of the network (figure 3). Using BatchNormalization, we had a normalized input in the next layers, but combined with ReLU activation function, it increases the sparsity of the network. We decided then to adopt different activation functions: LeakyReLU instead of ReLU and tanh on the first hidden layer. LeakyReLU can mitigate this issue of losing information in the case of negative values in input, while tanh takes all values between minus infinity to plus infinity, so it will always produce an output between plus 1 and minus 1. Tanh can though highlight the vanishing gradient problem: since this is a light architecture (few layers) with shirt skip connections and LeakyReLU, this becomes negligible.

### *G. AttackNet v2.2*

In this architectural version of our model (figure 4) we changed the way on how the skip connection are implemented compared to version 2.1: we utilized the add function of Keras instead of concatenate, and the same number of filters for each convolutional layer. This was done to create a similar architecture with faster inference times.

## V. RESULTS

In the following Table VI 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.

It can be seen from the table of results that for Replay-Att, CSMAD (RGB), 3DMAD and MS-Spoof datasets, each of the architectures is able to achieve close to ideal accuracy. Such accuracy is possible only with the selection of optimal hyperparameters.

From Fig. 1, 2 it can be seen that the optimal selection of the learning rate can significantly improve the result. Thus, AttackNet v2.2 reaches val_acc=1.0 on the CSMAD (RGB) dataset (the model makes correct predictions for all images from the test data) at learning rate=1e-03, while at learning rate=1e-05 the model is retrained and reaches val_acc=0.86. It should be noted that in Fig. 2 we did not manage to completely get rid of overfitting.

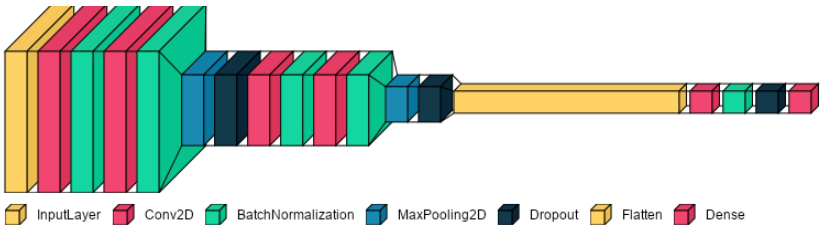


Fig. 1. LivenessNet

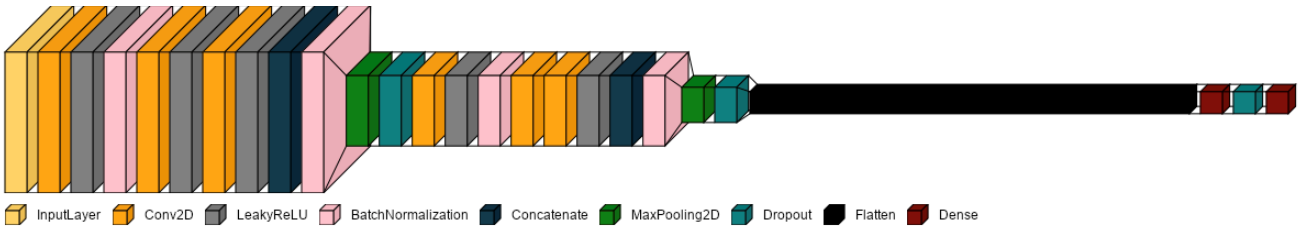


Fig. 2. AttackNet v1

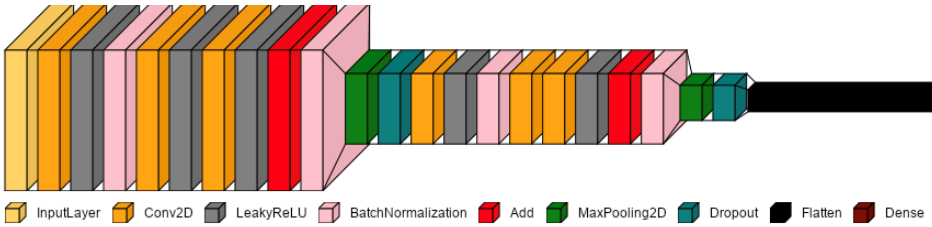


Fig. 3. AttackNet v2.1

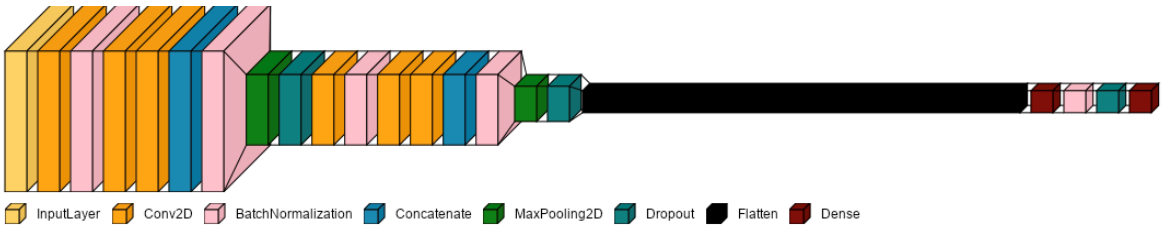


Fig. 4. AttackNet v2.2

TABLE VI. RESULTS OBTAINED FOR DIFFERENT DEEP LEARNING MODELS

| Network | **Dataset** | **Opt.** | **Mom** | **LR** | **Epochs** | **BS** | **Acc** | **Prec B/A** | | **Recall B/A** | | **F1 B/A** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LivenesNet | Our Dataset | Adam | - | 0.05 | 8 | 8 | 0.86 | 0.84 | 0.87 | 0.87 | 0.84 | 0.85 | 0.86 |
| AttackNet V1 | Our Dataset | Adam | - | 0.05 | 18 | 8 | 0.86 | 0.88 | 0.84 | 0.85 | 0.87 | 0.86 | 0.86 |
| AttackNet V2.1 | Our Dataset | Adam | - | 1E-03 | 63 | 8 | 0.89 | 0.96 | 0.83 | 0.85 | 0.95 | 0.9 | 0.88 |
| AttackNet V2.2 | Our Dataset | Adam | - | 0.05 | 22 | 8 | 0.89 | 0.89 | 0.88 | 0.88 | 0.89 | 0.89 | 0.89 |
| LivenessNet | Replay-Att | Adam | - | 1E-03 | 12 | 32 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| AttackNet V1 | Replay-Att | Adam | - | 1E-03 | 12 | 32 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| AttackNet V2.1 | Replay-Att | Adam | - | 1E-03 | 9 | 32 | 1.0 | 0.99 | 1.0 | 1.0 | 0.99 | 1.0 | 1.0 |
| AttackNet V2.2 | Replay-Att | Adam | - | 1E-03 | 20 | 32 | 1.0 | 0.99 | 1.0 | 1.0 | 0.99 | 1.0 | 1.0 |
| LivenesNet | CSMAD(RGB) | Adam | - | 1E-03 | 26 | 32 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| AttackNet V1 | CSMAD(RGB) | Adam | - | 1E-03 | 12 | 32 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| AttackNet V2.1 | CSMAD(RGB) | Adam | - | 1E-03 | 19 | 32 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| AttackNet V2.2 | CSMAD(RGB) | Adam | - | 1E-03 | 12 | 32 | 1.0 | 0.99 | 1.0 | 1.0 | 0.99 | 1.0 | 1.0 |
| LivenesNet | 3DMAD | Adam | - | 1E-03 | 98 | 32 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| AttackNet V1 | 3DMAD | Adam | - | 1E-03 | 85 | 32 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| AttackNet V2.1 | 3DMAD | SGD | 0.5 | 1E-03 | 15 | 32 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| AttackNet V2.2 | 3DMAD | SGD | 0.5 | 1E-03 | 15 | 32 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| LivenesNet | MS-Spoof | Adam | - | 1E-03 | 45 | 32 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| AttackNet V1 | MS-Spoof | Adam | - | 1E-03 | 20 | 32 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| AttackNet V2.1 | MS-Spoof | Adam | - | 1E-03 | 15 | 32 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| AttackNet V2.2 | MS-Spoof | Adam | - | 1E-03 | 42 | 32 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |

We managed to achieve a significant improvement in the efficiency of all models due to hyperparameter tuning. Our dataset turned out to be the most challenging dataset, on which the AttackNet v2.1 and AttackNet v2.2 models achieved the best result with a result of 89% accuracy on the validation data.

The optimal optimizer for Replay-Att, CSMAD(RGB), 3DMAD and MS-Spoof datasets is Adam with LR=1e-03. Loss under such conditions monotonically decreases, while val_loss jumps sharply from small to large values and back, as illustrated in Fig. 3.

A high learning rate in this case helps to cope with overfitting. Thus, with learning rates of 1e-04, 1e-05, 1e-06, val_loss smoothly decreases until a certain epoch, but after that, overfitting occurs and the decrease in val_loss stops, while train_loss and train_acc continue to decrease, which is illustrated in Fig. 4, 5.

Therefore to achieve best results and to prevent overfitting we used technique called early stopping. As mentioned earlier, starting from a certain epoch, the model begins to overfit and then does not reach the previous performance. It is at the epoch at which val_acc reaches its maximum values that we stop training, preventing the model from overftting.

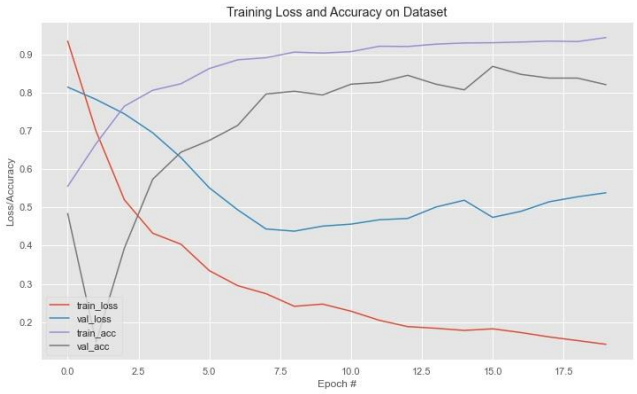


Fig. 5. CSMAD(RGB), AttackNet v2.2, LR=1e-05, opt=Adam

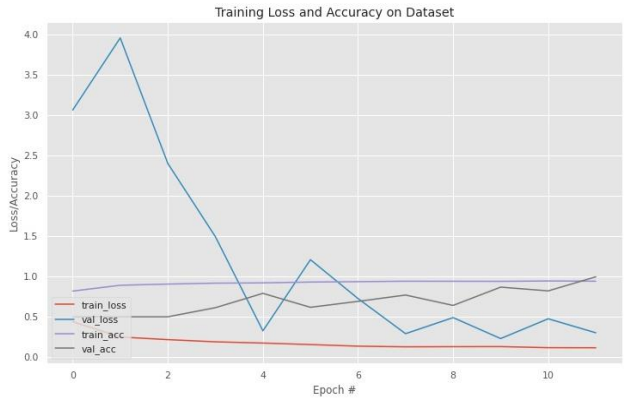


Fig. 6. CSMAD (RGB), AttackNet v2.2, LR=1e-03, opt=Adam


Fig. 7. CSMAD (RGB) AttackNet v2.1, opt=Adam, LR=1E-03.

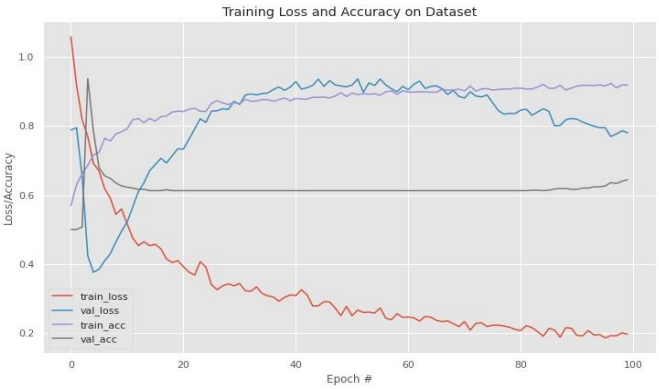


Fig. 8. CSMAD (RGB), LivenessNet, Adam, LR=1e-06, epochs=100

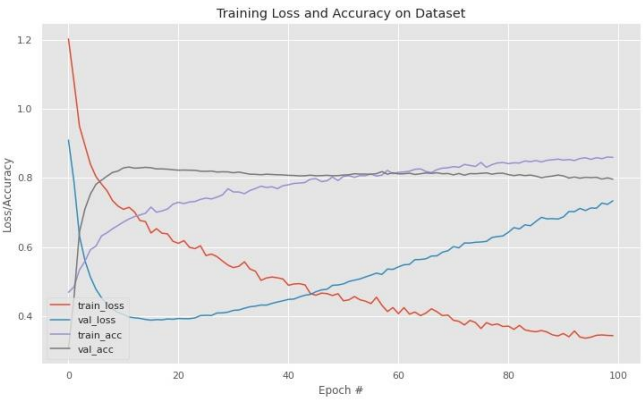


Fig. 9. Our, AttackNet v2.2, Adam, LR=1e-05, epochs=100

## VI. Discussion

The new model created is a good improvement over the initial architecture proposed. There aren't many known papers that used that kind of a approach with a total usage of a CNN architecture to obtain an Anti-Spoofing Attack system, in fact the not perfect accuracy that the model gets (in all of it's forms) suggest that a mixed approach is recommended. Evidence from results are that the addition on the fourth layer of the image depth it surely helps the Network to learn important feature. In general we think that current dataset available are too small to train CNNs from scratch. In fact most of the works that involves using a CNN for Anti Spoofing task are using existing pre trained Face Recognition CNN using transfer learning.

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 0.02-0.05 ms 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 particular features.

## VII. Conclusion

During the work, we trained and used different variations of deep neural networks to solve the task of liveness detection on five different datasets. On some datasets, the obtained results are close to ideal. For example, we managed to achieve almost 100% accuracy for various datasets and this allows us to assert the high efficiency of deep learning models for detecting fakes.

The convolutional network architecture proposed in [8] was used as the initial model. Our goal was to expand the options for applying deep learning to various attack scenarios. In particular, we investigated various attacks that were simulated by the corresponding datasets. Various methods were used to improve the detection results, including: changing the architecture of the neural network, selecting the best learning rate for the selected model and dataset, and other parameters. The proposed improvements significantly increase the detection efficiency.

It should also be noted that we presented a lighter architecture for PAD compared to the most commonly used models, which are deep convolutional neural networks. We have improved the generalization performance compared to the original architecture. This is important for detecting attacks in real time. In addition, there is an opportunity to implement our model on cheap equipment.

Future work may be to combine or utilize different dataset during the training phase, given the fact that the database shown in this work are old and possibly outdated. Another interesting point could be to investigate on how the different network layers are extracting valuable feature from the images and give meaning to them. Finally, could be important to extend the network with more layers and try to define a new successful convolutional neural network architecture.

## References


[1] G. Marcus and E. Davis, *Rebooting AI: Building Artificial Intelligence We Can Trust*. Vintage, 2019.

[2] K. A. Gates, *Our Biometric Future: Facial Recognition Technology and the Culture of Surveillance*. NYU Press, 2011.

[3] T. C. K. S. Sun, *Artificial General Intelligence: Executive Briefing on Facial Recognition Technology*.

[4] A. Kuznetsov, I. Oleshko, K. Chernov, M. Bagmut, and T. Smirnova, "Biometric Authentication Using Convolutional Neural Networks," in *Advances in Information and Communication Technology and Systems*, Cham, 2021, pp. 85–98. doi: 10.1007/978-3-030-58359-0_6.

[5] A. Kuznetsov, S. Fedotov, and M. Bagmut, "Convolutional Neural Networks to Protect Against Spoofing Attacks on Biometric Face Authentication," in *Digital Transformation*, Cham, 2021, pp. 123–146. doi: 10.1007/978-3-030-85893-3_9.

[6] G. Hua, "Facial Recognition Technologies," in *Encyclopedia of Big Data*, L. A. Schintler and C. L. McNeely, Eds. Cham: Springer International Publishing, 2022, pp. 475–479. doi: 10.1007/978-3-319-32010-6_93.

[7] C. Libby and J. Ehrenfeld, "Facial Recognition Technology in 2021: Masks, Bias, and the Future of Healthcare," *J Med Syst*, vol. 45, no. 4, p. 39, Feb. 2021, doi: 10.1007/s10916-021-01723-w.

[8] "Liveness Detection with OpenCV," *PyImageSearch*, Mar. 11, 2019. https://www.pyimagesearch.com/2019/03/11/liveness-detection-with-opencv/ (accessed Aug. 17, 2022).

[9] Y. A. Ur Rehman, L. M. Po, and M. Liu, "Deep learning for face anti-spoofing: An end-to-end approach," in *2017 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*, Sep. 2017, pp. 195–200. doi: 10.23919/SPA.2017.8166863.

[10] K. Mallat and J.-L. Dugelay, "Indirect synthetic attack on thermal face biometric systems via visible-to-thermal spectrum conversion," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jun. 2021, pp. 1435–1443. doi: 10.1109/CVPRW53098.2021.00159.

[11] K. Kotwal and S. Marcel, "CNN Patch Pooling for Detecting 3D Mask Presentation Attacks in NIR," in *2020 IEEE International Conference on Image Processing (ICIP)*, 2020, pp. 1336–1340. doi: 10.1109/ICIP40778.2020.9191240.

[12] S. Jia, G. Guo, and Z. Xu, "A survey on 3D mask presentation attack detection and countermeasures," *Pattern Recognition*, vol. 98, p. 107032, Feb. 2020, doi: 10.1016/j.patcog.2019.107032.

[13] G. Wang, Z. Wang, K. Jiang, B. Huang, Z. He, and R. Hu, "Silicone mask face anti-spoofing detection based on visual saliency and facial motion," *Neurocomputing*, vol. 458, pp. 416–427, Oct. 2021, doi: 10.1016/j.neucom.2021.06.033.

[14] C. Nagpal and S. R. Dubey, "A Performance Evaluation of Convolutional Neural Networks for Face Anti Spoofing." arXiv, Mar. 27, 2019. doi: 10.48550/arXiv.1805.04176.

[15] S. Bhattacharjee, A. Mohammadi, and S. Marcel, "Spoofing Deep Face Recognition with Custom Silicone Masks," in *2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, 2018, pp. 1–7. doi: 10.1109/BTAS.2018.8698550.

[16] "Custom Silicone Mask Attack Dataset (CSMAD)," *Idiap Research Institute, Artificial Intelligence for Society*. https://www.idiap.ch:/en/dataset/csmad/index_html (accessed Aug. 17, 2022).

[17] N. Erdogmus and S. Marcel, "Spoofing in 2D face recognition with 3D masks and anti-spoofing with Kinect," in *2013 IEEE Sixth International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, Sep. 2013, pp. 1–6. doi: 10.1109/BTAS.2013.6712688.

[18] "3DMAD," *Idiap Research Institute, Artificial Intelligence for Society*. https://www.idiap.ch:/en/dataset/3dmad/index_html (accessed Aug. 17, 2022).

[19] I. Chingovska, N. Erdogmus, A. Anjos, and S. Marcel, "Face Recognition Systems Under Spoofing Attacks," in *Face Recognition Across the Imaging Spectrum*, T. Bourlai, Ed. Cham: Springer International Publishing, 2016, pp. 165–194. doi: 10.1007/978-3-319-28501-6_8.

[20] "Multispectral-Spoof (MSSpoof)," *Idiap Research Institute, Artificial Intelligence for Society*. https://www.idiap.ch:/en/dataset/msspoof/index_html (accessed Aug. 17, 2022).

[21] I. Chingovska, A. Anjos, and S. Marcel, "On the effectiveness of local binary patterns in face anti-spoofing," in *2012 BIOSIG - Proceedings of the International Conference of Biometrics Special Interest Group (BIOSIG)*, Sep. 2012, pp. 1–7.

[22] "Replay-Attack," *Idiap Research Institute, Artificial Intelligence for Society*. https://www.idiap.ch:/en/dataset/replayattack/index_html (accessed Aug. 17, 2022).

[23] D. Soydaner, "A Comparison of Optimization Algorithms for Deep Learning," *Int. J. Patt. Recogn. Artif. Intell.*, vol. 34, no. 13, p. 2052013, Dec. 2020, doi: 10.1142/S0218001420520138.

[24] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization." arXiv, Jan. 29, 2017. doi: 10.48550/arXiv.1412.6980.