Spoof Detection with RGB Cameras in Facial Recognition Systems

Ye Li, Kairen Ye, Jiyao Yuan
{liye5, kairenye, yuan999}@stanford.edu

***Abstract*—As face has gained much attention as a promising mode of identity, a robust face anti-spoof algorithm is essential to massively deploying facial recognition as the primary mode of authentication. Instead of relying on more expensive depth sensors, we propose to implement a deep-learning based solution to detect spoof images with regular RGB cameras. Our model will learn from two auxiliary supervisions (pseudo-depth maps and rPPG signals) to discern live vs spoof images. We show that our CNN-RNN architecture can achieve a recall rate close to 100% for spoof faces, which meets the rigorous requirements by biometric authentication systems.**

I. Introduction

With the increasing application of biometric authentication, face has gained much attention as a promising mode of identity. Consequently, a robust anti-spoof algorithm can tremendously boost users and stakeholders confidence in massively deploying facial recognition as the primary mode of authentication. In previous traditional anti-spoofing algorithms, binary supervision is commonly used to train deep neural network (DNN), but it lacks explainability and has poor generalization [1]. Therefore, a novel DNN with auxiliary supervisions is proposed. In this novel method, the model would estimate pseudo-depth maps without the need for depth input from the camera and also estimate remote photoplethysmography (rPPG) signal, which encodes heart rate information, for discerning live versus spoof images. With this innovative and robust approach, facial authentication using ubiquitous 2D RGB cameras can be deployed with confidence.

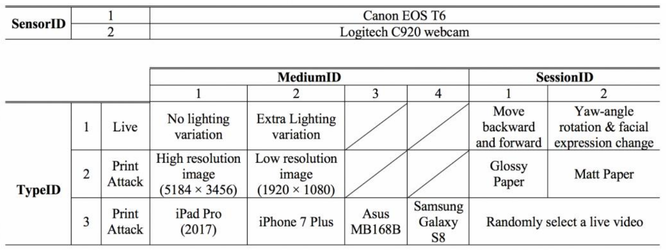
II. Related Work

In previously published work on face anti-spoof, researchers use similar input image formats but instead adopt binary supervision (i.e. 0 for spoof and 1 for live) with softmax loss to train CNN models [7, 8]. However, the pitfall of these approaches is that training such networks result in poor generalization, because binary supervision doesn't enforce the network to learn image degradation characteristics that might indicate faithful spoof patterns [1]. Rather, these networks might learn arbitrary cues such as screen bezels [1]. Moreover, these approaches do not provide explainable outputs.

In our work, we refer to [1] and introduce a novel DNN that is trained on 2D pseudo-depth maps and rPPG signals. Then, classification of spoof versus live is performed by thresholding the sum of the norms of the outputs. As opposed to a completely end-to-end approach with binary supervision, this novel DNN achieves better generalization and more stable results.

III. Dataset and Features

The dataset contains 165 subjects [1]. Each subject has approximately 8 live video clips and approximately 20 spoof video clips. All videos are about 15 seconds in length with HD 1080P resolution recorded at 30 FPS. Live videos are taken with variations of illumination, distance, pose, and facial expression. Spoof videos are taken with different attacks, such as paper with various textures and tablet devices of different brands. In total, the dataset contains 4,478 video files, occupying approximately 211 GB on disk. Fig. 1 illustrates all the possible combinations in the dataset as well as the naming convention of the video files. By inspecting the dataset, we were able to infer the exact train-test split used in the original paper [1]. We would use the same train-test split, where the training set contains 90 subjects, and the test set contains the rest 75 subjects.



SensorID	1		Canon EOS T6					
	2		Logitech C920 webcam					

			MediumID				**SessionID**	
			1	2	3	4	1	2
TypeID	1	Live	No lighting variation	Extra Lighting variation			Move backward and forward	Yaw-angle rotation & facial expression change
	2	Print Attack	High resolution image (5184 × 3456)	Low resolution image (1920 × 1080)			Glossy Paper	Matt Paper
	3	Print Attack	iPad Pro (2017)	iPhone 7 Plus	Asus MB168B	Samsung Galaxy S8	Randomly select a live video	

Fig. 1: All the possible combinations in the dataset [1]. The naming convention of the video files follows the SubjectID_SensorID_TypeID_MediumID_SessionID.mov format.

IV. Methods

Since the reference paper [1] does not release its open-source implementation, we proposed to implement similar approaches and models from scratch in PyTorch and then compared our results to the original paper. We devoted significant efforts to label our data and generate the auxiliary supervisions to be used in training, which are the pseudo-depth maps acquired from 3D face alignment and the rPPG signals encoding heart rate information. We then created a custom non-rigid registration layer that performs face frontalization for more consistent model learning. Finally, we built the network with a CNN-RNN architecture that learns from both spatial and temporal cues.

A. CNN-RNN Architecture

Our DNN architecture is illustrated in Fig. 2. The network utilizes both convolutional and recurrent layers, and we use an end-to-end scheme to train the network with the two loss

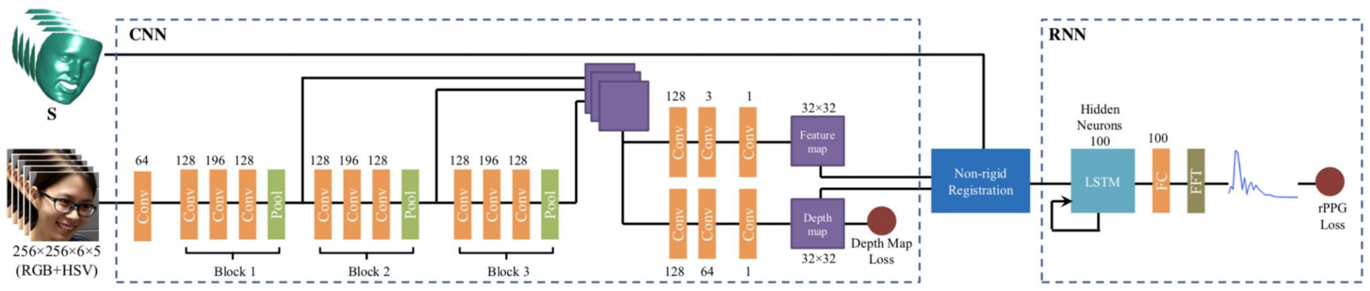


Fig. 2: The overall network architecture of the anti-spoof model. The network contains both convolutional and recurrent layers and has two loss functions (one for depth map and the other for rPPG signal) [1].

functions [1]. The convolutional layers help the network capture cues for generating the pseudo-depth from a single frame of image, whereas the recurrent components help the network capture the temporal information necessary for learning the rPPG signal across multiple image frames over time. The Depth Map Loss is defined as

$$\Theta_D = \underset{\Theta_D}{\operatorname{argmin}} \sum_{i=1}^{N_d} \|\mathrm{CNN}_D(\mathbf{I}_i; \Theta_D) - \mathbf{D}_i\|_1^2, \tag{1}$$

where Θ_D is the CNN parameters, N_d is the number of training images, $\mathbf{I}_i \in \mathbb{R}^{256\times256}$ is the i^{th} input frame, and $\mathbf{D}_i \in \mathbb{R}^{32\times32}$ is the ground truth depth map for the i^{th} training image [1]. The rPPG Loss is defined as

$$\Theta_R = \underset{\Theta_R}{\operatorname{argmin}} \sum_{i=1}^{N_s} \|\mathrm{RNN}_R([\{\mathbf{F}_j\}_{j=1}^{N_f}]_i; \Theta_R) - \mathbf{f}_i\|_1^2, \tag{2}$$

where Θ_R is the RNN parameters, $\mathbf{F}_j \in \mathbb{R}^{32\times32}$ is the frontalized feature map, N_s is the number of sequences, and $\mathbf{f}_i \in \mathbb{R}^{50}$ is the ground truth rPPG signal for the i^{th} training image [1].

B. Non-rigid registration layer

A non-rigid registration layer exists between the convolutional and the recurrent portions of the network. As shown in Fig. 3, the layer takes the feature map $\mathbf{T}$, the depth map $\mathbf{D}$, and the 3D shape $\mathbf{S}$ as inputs and outputs $\mathbf{F}$ to the recurrent part of the network. Additionally, we have a predefined 3D shape $\mathbf{S}_0$, which is a frontal 3D face shape. For $\mathbf{T}$, $\mathbf{D}$, $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{F}$, the dimension is $\mathbb{R}^{32\times32}$. For $\mathbf{S}$ and $\mathbf{S}_0$, the dimension is $\mathbb{R}^{3\times53215}$.

First, we threshold $\mathbf{D}$ to generate a binary mask $\mathbf{V}$. Then, we take the element-wise product between the binary mask $\mathbf{V}$ and $\mathbf{T}$ and denote the result as $\mathbf{U}$. Then, two mapping functions, f_1 and f_2, are used to frontalize $\mathbf{U}$ to $\mathbf{F}$. Function f_1 maps an index of column vectors of $\mathbf{S}$ to index (i_*, j_*) of $\mathbf{U}$. Function f_2 maps index (i, j) of $\mathbf{F}$ to an index of column vectors of $\mathbf{S}_0$. Mathematically, each element of $\mathbf{F}$ is computed as follows:

$$\mathbf{F}(i,j) = \mathbf{U}(i_*, j_*), \tag{3}$$

$$i_*, j_* = f_1(f_2(i,j)). \tag{4}$$

To demonstrate our implementation of the face frontalization step in the non-rigid registration layer, we take a 2D depth map as $\mathbf{U}$ and frontalize it to $\mathbf{F}$, as seen in Fig. 3 where the bottom left image is the input 2D depth map and the bottom right image is the frontalized depth map.

The non-rigid registration layer has three main advantages [1]:

- the input data are aligned (i.e. frontalized) and the RNN can compare the feature maps without adjusting for variations in pose;
- backgrounds are removed;
- the binary mask weakens the feature map activations of spoof faces.

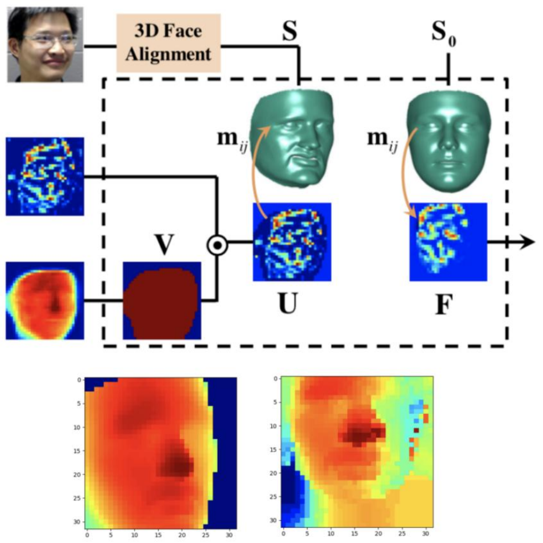


Fig. 3: Non-rigid registration illustration (top) and an example of our implementation (bottom).

C. Generating Supervision Data

1) Face detection and cropping: In the SiW dataset, each video has an associated $.face$ file, which includes the coordinates of face bounding boxes in every frame. If no faces are detected in a frame, the coordinates will be zero. When processing and preparing our training data, we extract the

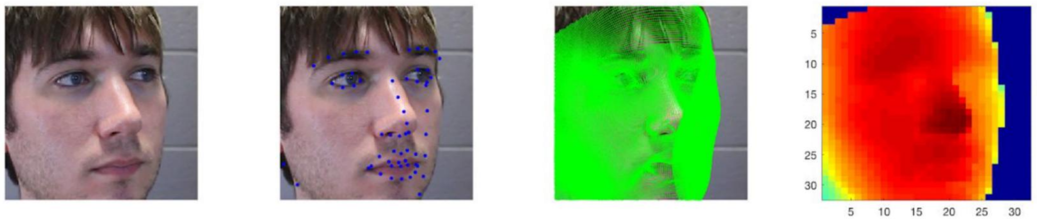


Fig. 4: From left to right, each picture shows 1) the original face image cropped by given coordinates, 2) fitting the sparse 68 landmarks onto the face image, 3) fitting the dense 53,215 vertices onto the face image using DeFA, 4) the $\mathbb{R}^{32\times32}$ depth map by applying Z-Buffer to the 53,215 vertices.

first 60 frames with valid face coordinates, crop each frame according to the face bounding box coordinates, and output the cropped frames as the data we use during training for that video ID. Because some frames might not have a detected face and hence might not have valid bounding box coordinates, the 60 frames we extracted might not be strictly consecutive in 30 fps, which could slightly distort our generated rPPG supervision. However, such variation does not significantly impact training integrity, since slight randomness could help prevent overfitting.

2) Estimating 3D Face Shape with Dense Face Alignment (DeFA): One of the two auxiliary supervisions used in training our network is the pseudo-depth map. To estimate the depth map for a 2D face image, the dense face alignment (DeFA) approach is used [3]. First of all, the frontal dense 3D shape $\mathbf{S}_F \in \mathbb{R}^{3\times53215}$ (i.e. the dense 3D face shape is estimated with 53,215 vertices) is represented as a linear combination of the identity bases and expression bases

$$\mathbf{S}_F = \mathbf{S}_0 + \sum_{i=1}^{N_{id}} \alpha_{id}^i \mathbf{S}_{id}^i + \sum_{i=1}^{N_{exp}} \alpha_{exp}^i \mathbf{S}_{exp}^i, \tag{5}$$

where $\alpha_{id} \in \mathbb{R}^{199}$ and $\alpha_{exp} \in \mathbb{R}^{29}$ are the identity and expression parameters [3]. We use the Basel 3D face model [4] and the facewarehouse [5] as the identity and expression bases. Then, with the estimated pose parameters $\mathbf{P} = (s, \mathbf{R}, \mathbf{t})$, where $\mathbf{R}$ is a 3D rotation matrix, $\mathbf{t}$ is a 3D translation, and s is a scaling factor, the 3D shape $\mathbf{S}$ is aligned to the 2D face image via

$$\mathbf{S} = s\mathbf{R}\mathbf{S}_F + \mathbf{t}. \tag{6}$$

The rotation matrix and the shape parameters (i.e. α_{id} and α_{exp}) for aligning to a specific 2D face image are acquired from the output of running a forward pass of the pretrained DeFA convolutional network, obtained from the author's open-source DeFA implementation [3]. In Fig. 4, the result of running DeFA on a cropped face image is shown, where the third picture from the left shows the result of projecting the 53,215 vertices onto the original 2D face image.

3) 2D Depth Map Generation with Z-Buffer: As seen above in *Section C.(2)*, DeFA outputs $\mathbf{S} \in \mathbb{R}^{3\times53215}$, which corresponds to the dense 53,215 3D vertices fitting a persons face image. However, we still need to transform these dense points into a depth map $\mathbf{D} \in \mathbb{R}^{32\times32}$, which is one of the two supervisions needed in training [1]. To do so, we first normalize the $\mathbf{z}$ values of the 3D vertices from DeFA to be within [0,1]. Then, we apply the Z-Buffer algorithm to project these 3D vertices onto a $\mathbb{R}^{32\times32}$ down-sampled depth map. In the Z-Buffer algorithm, for each point on the $\mathbb{R}^{32\times32}$ depth map, we look at all 64 corresponding vertices in the original $\mathbb{R}^{256}$ image dimension and select the largest $\mathbf{z}$ value to be put on the 32x32 depth map.

4) rPPG Signal Generation from Consecutive Video Frames: rPPG is the technique of measuring heart rates without any physical contact with human skin and has gained attention as a potential indicator of spoofing. rPPG signal is related to the intensity changes of facial skin over time, which are highly correlated with blood flow [1]. In the labeling process, we first track a region on the forehead to be used later for computing the rPPG signal [1]. Then, we use chrominance-based rPPG, which computes color difference to eliminate the specular reflection and estimate two orthogonal chrominance signals [1]. We apply a band-pass filter to the chrominance signals and apply the Fast Fourier Transform (FFT) to get the final rPPG signal. Mathematically, for the tracked region on the subjects forehead, we compute

$$\mathbf{x}_f = 3\mathbf{r}_f - 2\mathbf{g}_f, \tag{7}$$

$$\mathbf{y}_f = 1.5\mathbf{r}_f + \mathbf{g}_f - 1.5\mathbf{b}_f, \tag{8}$$

where $\mathbf{r}_f$, $\mathbf{g}_f$, and $\mathbf{b}_f \in \mathbb{R}^{100}$ has each individual value in the vector computed from the average of the respective color channel at a single video frame [6]. The three vectors are then normalized and band-pass filtered to 0.6-4Hz (36-240 bpm). Finally, the ratio of the standard deviations of the chrominance signals

$$\gamma = \frac{\sigma(\mathbf{x}_f)}{\sigma(\mathbf{y}_f)} \tag{9}$$

is used to calculate signal $\mathbf{p}$

$$\mathbf{p} = 3(1 - \frac{\gamma}{2})\mathbf{r}_f - 2(1 + \frac{\gamma}{2})\mathbf{g}_f + \frac{3\gamma}{2}\mathbf{b}_f. \tag{10}$$

Finally, the ground truth rPPG signal $\mathbf{f} \in \mathbb{R}^{50}$ is obtained by applying FFT to $\mathbf{p}$ [1]. Note that during training, for each subject's live data, only one ground truth rPPG signal is used, because we assume that the videos of the same subject under different pose, illumination, and expression (PIE) variations have the same ground truth rPPG signal, which is valid since the data for the same subject were collected in a short span of time ($<$ 5 minutes) [1]. In Fig. 5, we show an example of an rPPG signal we have computed on a 5-second video footage. With the video recorded at 20 FPS, we used 100 frames (i.e. 5 seconds of footage). We consistently tracked a region on the subject's forehead and applied the steps outlined above to acquire the final signal. Note that the peak of the example is at 0.8 Hz, which is quite reasonable since this corresponds to a heart rate of 0.860=48 bpm.

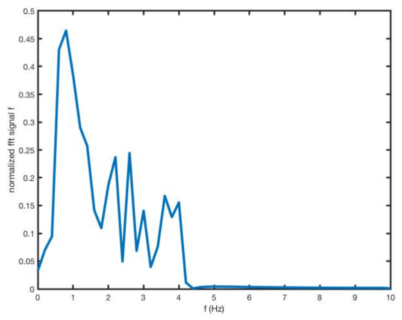


Fig. 5: An example of rPPG signal.

V. Experiments

A. Baseline model with SVM

Due to its easy implementation and relatively short training time, SVM is chosen as our baseline model to classify the input RGB face images into live versus spoof.

Unlike our CNN-RNN approach, SVM model is trained with binary supervision as it cannot be designed to estimate 2D depth map and rPPG signal. Each input image is a video frame in the SiW dataset. Specifically, a total of 3,112 images exist in the dataset used in the baseline SVM model, where 1,616 are live images and 1,496 are spoof images. Data are then split into 80% for training and 20% for validation.

We first detect and crop the face image in each sample. Each face image is then resized into a 32×32 RGB image, with INTER_AREA as the interpolation type. We then normalize and flatten samples into a $\mathbb{R}^{3072}$ vector. Note that when the length of vector is larger, the variance would be larger, because more features are taken into account. On the other hand, when the length of vector is smaller, the bias would be larger, because more information would be lost during the process of flattening an image into a vector. In our experiment, an $\mathbb{R}^{3072}$ vector results in acceptable variance, bias, and running time. Finally, the vector is fed into an SVM (C = 0.1) with degree-two polynomial kernel.

B. Training the CNN-RNN model

We trained our CNN-RNN model in an end-to-end manner by adding the two loss functions (1) and (2) into an overall loss, from which the model backpropagates. During backpropagation, all the layers' parameters get updated due to the use of the overall loss. Since the two losses differ by at least four orders of magnitude, we put a hyperparameter μ, which serves as a weight on the depth map loss in the overall loss calculation, for better training results.

We concatenated HSV channels to each RGB image frame as input. We used a sequence length of 5 for the LSTM layers. Hence, the input dimension for CNN is at least $5 \times 6 \times 256 \times 256$ ($batch \times channel \times H \times W$). After forward propagating through the CNN layers and the non-rigid registration layer, the feature map output will be reshaped to $1 \times 5 \times 1024$ ($batch \times sequence \times input\_length$) since every five consecutive images will generate one rPPG signal.

Finally, we used a mini-batch size of 10 for CNN (input dimension is $10 \times 6 \times 256 \times 256$), and a mini-batch size of 2 for RNN (input dimension is $2 \times 5 \times 1024$). The train-test split is approximately 60%-40%. We used a random yet balanced subset of the full training data due to time and resource constraints and trained for 10 epochs, where each epoch is 4440 steps. See Fig. 7 for our training losses.

C. Classification score at test time

At test time, after generating the output depth maps and rPPG signals, we computed a score on each test sample using the formula

$$score = \|\mathbf{f}\|_2^2 + \lambda \|\mathbf{D}\|_F^2 \tag{11}$$

where $\mathbf{D}$ is the network's output depth map of the last frame and $\mathbf{f}$ is the network's output rPPG signal. The weight λ is set to be 0.015 [1]. The threshold on the score for classifying live vs. spoof is 6.3. Using the aforementioned λ and threshold, we calculated accuracy, precision, and recall. In the context of detecting spoof faces in facial authentication systems, the objective is to get approximately 100% recall for spoof faces, while still achieving relatively high accuracy and precision.

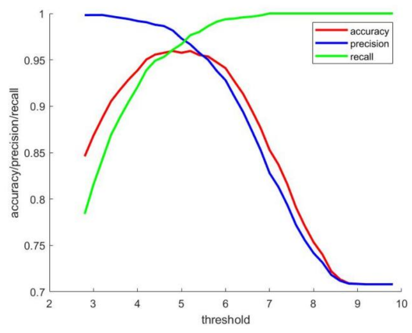


Fig. 6: Effects on tuning threshold value on validation accuracy/precision/recall.

D. Hyperparameter tuning

For the majority of our hyperparameter choices, such as mini-batch size and number of epochs, we deferred to [1]. However, we tuned the loss weight μ and the evaluation threshold, which are discussed below.

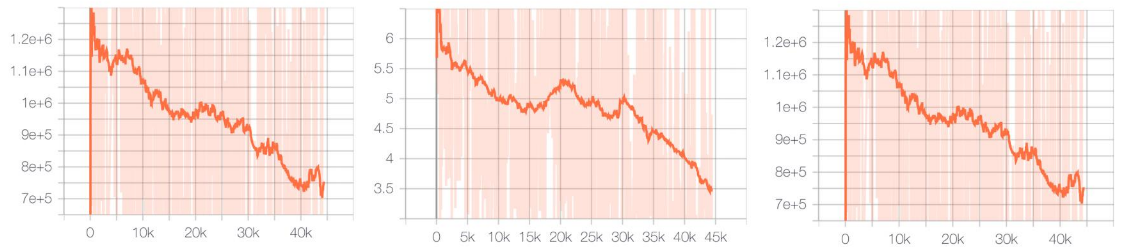


Fig. 7: From left to right (with TensorBoard smoothing factor 0.99): 1) depth map loss, 2) rPPG loss, 3) overall loss.

1) Weight when combining the two losses: As mentioned above, since our two losses differ by many orders of magnitude, we proposed to include a hyperparameter μ when calculating the total loss. We tried setting $\mu = \frac{1}{10000}$ at first since that is approximately the order-of-magnitude difference between the dimensions of the two losses. However, the total loss did not converge faster. For training effectiveness, we finally set $\mu = 1$.

2) Classification threshold at test time: After calculating the score of each test sample, we tuned the threshold value for classification and observed how validation accuracy, precision, and recall changed with the threshold. According to Fig. 8, we observe a tradeoff between recall and precision, while the accuracy first increases and then decreases. Although we achieved the highest accuracy 95.96% when the threshold is set around 5.1, in the interest of robustly detecting spoof faces in facial authentication systems, a recall close to 100% is desired. Therefore, we used a threshold of 6.3 in our evaluation.

VI. Results & Discussion

Example depth map and rPPG outputs from our trained network when given a real and a spoof image are shown in Fig. 8. The output depth map and rPPG signal of a live face image have significant nonzero values, resulting in a high score as defined by Eq. (11). On the contrary, those of a spoof face image have values mostly close to zero, resulting in a score close to 0.

Table 1 shows the validation accuracy, precision, and recall for the baseline SVM model versus our trained CNN-RNN model. Note that these metrics are computed by considering spoof prediction as positive results. Recall that the motivation behind our work is to create a robust and reliable face antispoof system so that facial authentication systems can be deployed with confidence. With this objective, high recall (i.e. the proportion of actual spoof faces that were identified correctly) is of the utmost importance. Although our model achieves relatively similar accuracy and precision as the baseline SVM model, the trained CNN-RNN model's recall is significantly higher (99.52%). In fact, we were able to achieve 100% recall on the test set if we set the classification score threshold higher. Therefore, this model, which is trained to output auxiliary information instead of mere binary outputs, also carries the significant benefit of flexibility during deployment. For example, if the application is less stringent on detecting spoof faces but values detection accuracy, the threshold could be adjusted accordingly at test time without having to re-train the model.

	Validation Accuracy	Validation Precision	Validation Recall
SVM	91.80%	95.93%	86.62%
CNN-RNN	92.15%	90.37%	99.52%

TABLE I: Evaluation and Comparison

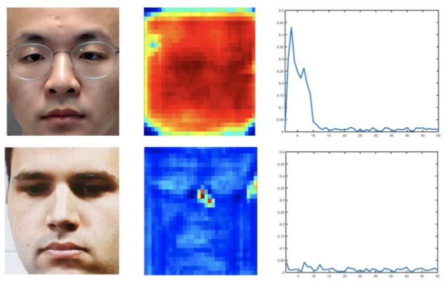


Fig. 8: Examples of output depth maps and rPPG signals from the network for live (top) and spoof (bottom) images.

VII. Conclusion & Future Work

Our trained CNN-RNN model with auxiliary supervision performs well with a high recall rate. This aligns with the rigorous antispoof requirements by facial authentication systems.

In the future, we aim to optimize our process and model in a number of ways. We would pipeline the data-loading in our training program so that each mini-batch would take less time. We would instead train on all 2409 video samples should time permit. Finally, we would evaluate our model on diverse test sets to more realistically evaluate the performance of our model.

Contributions and Acknowledgement

This project is completed with the conscientious and co-operative effort of three group members Ye Li, Kairen Ye, and Jiyao Yuan evenly in terms of idea generation, model establishment, code deployment as well as report writing. We would like to thank Abhijeet Shenoi, Kian Katanforoosh, and Yaojie Liu for providing valuable feedback.

Repository link

https://github.com/kairenye/cs230_antispoof

(access granted to cs230-stanford)

References

[1] Y. Liu, A. Jourabloo, and X. Liu, Learning deep models for face anti-spoofing: Binary or auxiliary supervision, in CVPR, 2018.

[2] D. E. King. Dlib-ml: A machine learning toolkit. JMLR, 10(Jul):17551758, 2009.

[3] Y. Liu, A. Jourabloo, W. Ren, and X. Liu. Dense face alignment. In ICCVW, pages 16191628, 2017.

[4] P. Paysan, R. Knothe, B. Amberg, S. Romdhani, and T. Vetter. A 3D face model for pose and illumination invariant face recognition. In AVSS, pages 296301, 2009.

[5] C. Cao, Y. Weng, S. Zhou, Y. Tong, and K. Zhou. Facewarehouse: A 3D facial expression database for visual computing. IEEE Trans. Vis. Comput. Graphics, 20(3):413425, 2014.

[6] G. de Haan and V. Jeanne. Robust pulse rate from chrominance-based rPPG. IEEE Trans. Biomedical Engineering, 60(10):28782886, 2013.

[7] L. Feng, L.-M. Po, Y. Li, X. Xu, F. Yuan, T. C.-H. Cheung, and K.-W. Cheung. Integration of image quality and motion cues for face anti-spoofing: A neural network approach. J. Visual Communication and Image Representation, 38:451 460, 2016.

[8] L. Li, X. Feng, Z. Boulkenafet, Z. Xia, M. Li, and A. Ha- did. An original face anti-spoofing approach using partial convolutional neural network. In IPTA, 2016.