Real-time Faceliveness Detection with Deep Learning

Pongkorn Settasompop, *st121622@ait.asia* Pranisaa Charnparttaravanit, *st121720@ait.asia*
Chanapa Pananookooln, *st121395@ait.asia*

***Abstract*—Face recognition technology has developed rapidly in recent years. Traditionally, binary classification or Convolutional Neural Network by using feature representation extracted are the way to tackle the problems, but they often face the overfitting problem and their performance worsens due to the increasing number of wide variety of attacks. However, this problem was solved by reformulating face anti-spoofing under the assumption that live sample shares the same nature which can be categorized into one class and not-live samples are outliers. This, however, was not implemented as a real-time system. Therefore, this paper investigates FAS capabiltiy in real-time faceliveness detection. We also compared the classification accuracy of FAS to GoogLeNet, Pretrained GoogLeNet, ResNet18, Pretrained ResNet18 and SEResNet18 on our custom datasets which contain various types of spoof attacks. Despite unsatisfactory performance of FAS on classification problem when compared to GoogLeNet, ResNet18 and SEResNet18, it outperformed them once MTCNN and OpenCV were integrated to enable real-time ability.**

***Index Terms*—FAS, presentation attacks, residual learning,**

I. Introduction

FACE recognition technology has developed rapidly in recent years. It is direct, user friendly and convenient compared to other methods when it comes to authentication matters. However, this comes with a price as face recognition algorithms are not able to differentiate 'live' face from 'not live' face which is a major security issue. Hackers may fraud the systems by using facial pictures such as portrait photographs and 3D facial masks. In order to guard against such spoofing, face anti-spoofing (FAS) is developed.

Traditionally, binary classification by using feature representation extracted by the methods such as LBP, HOG and SIFT is usually a way to tackle such problems. On the other hand, some took a step further and employ Convolutional Neural Network (CNN) to extract relevant feature representations due to its ability to learn more discriminative features. Despite such ability, overfitting is a common problem researchers face which is due to the increasing number of wide variety of attack mediums present in the real world. However, the aforementioned problem was recently solved by reformulating FAS under the assumption that live samples share the same nature which can be categorized into one class and not-live samples are outliers of a closed-set and belong to an open-set[cite]. This effective separation between the live closed-set and the spoof open-set was enabled by formulating spoof cues as a feature map of the same size of the input image. The spoof cue map is prone to be a non-zero map for a spoof image while an all-zero map for a live image. This spoof cues are able to encode spatial information which are learnt by the residual-learning framework consisting of a spoof cue generator and an auxiliary classifier. However, the available codes are originally not written in PyTorch, this system is only able to perform FAS on image inputs where faces are carefully cropped, is not yet compatible with video inputs and real-time operating is still not an option.

Aiming at the aforementioned issues, this study aims to translate the reformulated FAS orignally written code into PyTorch, integrate Multi-Task Cascaded Convolutional Neural Networks (MTCNN) into the reformulated FAS [1] model to enable automatic cropping and to allow feasibility of video inputs and lastly, develop the system to be a real-time operating system. Furthermore, we also reported the comparison of classification performance of the formulated FAS system with several standard models namely GoogLeNet, Pretrained GoogLeNet, ResNet18, Pretrained ResNet18 and SEResNet18 on our custom datasets containing 4 different data sets of a wide variety of spoof types.

The comparison of classification accuracies of different models show that the reformulated FAS model performed unexpectedly lower than other models and achieved 70.17%. However, once integrated MTCNN and OpenCV to allow real-time detection ability, the reformualted FAS outperformed all previously mentioned models, both in terms of accuracy as well as the generalization ability.

The contributions of this work include (1) we developed a reformulated FAS in the version of PyTorch. (2) we developed auto-cropping PyTorch-reformulated-FAS. (3) we present a real-time FAS operating system.

II. Related Work

A. Generalized Spoof Cues for Face Anti-Spoofing

Feng et al. [1] reformulated FAS in an anomaly detection perspective and propose a residual-learning framework to learn the discriminative live-spoof differences defined as spoof cues. The framework consists of a spoof cue generator and an auxiliary classifier. These spoof cues are formulated as a feature map of the same size as the input image which are learnt by this residual-learning network. The spoof cue generator separates spoof cues for live and spoof samples by setting explicit regression loss for live samples to minimize the

magnitudes of their spoof cues and setting implicit constraints on the spoof samples. By doing so, the spoof map of spoof sample is prone to be a non-zero map while that of a live image is prone to be an all-zero map. Then, the spoof cue maps and the input image, which are skip-connected in a residual learning manner, are fed to an auxiliary classifier to further help learn more discriminative spoof cues.

Due to the wide variety of unseen spoofs, this study was motivated from the anomaly detection method. They assumed that the live samples belong to a closed set that forms a compact sphere in the learned feature representation space and the spoof samples belong to a open set that are outliers, far away from the center of the sphere.

Let the model input space be $(\mathcal{X} \subseteq R^d)$ and output space be $(\mathcal{Z} \subseteq R^p)$ and $\phi(\cdot;\mathcal{W}) : \mathcal{X} \to \mathcal{Z}$ which has L hidden layers and set of weights $\mathcal{W} = \{\mathcal{W}^1, ..., \mathcal{W}^{\mathcal{L}}\}$. Given $\mathcal{N}_l$ live samples $(x_1, ..., x_{\mathcal{N}_l} \subseteq \mathcal{X})$ and $\mathcal{N}_s$ spoof samples $(y_1, ..., y_{\mathcal{N}_s} \subseteq \mathcal{X})$, let c be the center of live samples in the output space $\mathcal{Z}$, the objective functions are

$$\min_{\mathcal{W}} \frac{1}{N_l} \sum_{i=1}^{N_l} \|\phi(x_i;\mathcal{W}) - c\|^2 \tag{1}$$

$$\max_{\mathcal{W}} \frac{1}{N_s} \sum_{i=1}^{N_s} \|\phi(y_i;\mathcal{W}) - c\|^2 \tag{2}$$

The residual learning framework is adopted from U-Net [2] architecture where there are skips connection from encoder to decoder at multiple scales and the spoof cues are output of the model. The Encoder E is adapted from ResNet18 pretrained from ImageNet. It contains 4 encoder residual blocks and end with a fully-connected layer followed by the decoder D. The docoder composed of 5 decoder residual blocks where the feature maps is upsampled by nearest-neighbour interpolation. The feature maps from the symmetric position from the encoder were concatenated with the corresponding feature maps from the decoder. With tanh applied to the last layer of decoder, the generated spoof cues with linearly normalized value of [-1, 1] are output from the model. The spoof cues are then added to the input images to feed into the auxiliary classifier for obtaining predicted labels which will be used in training. The auxiliary classifier is also a Pretrained ResNet18 with 4 residual blocks and cross entropy loss at the output.

Given RGB image I and spoof cue map C of the same size which is output from the spoof cue generator, C remains unknown for a spoof samples but for a live sample C is a zero map. Thus, the spoof cue regression loss is the pixel-wise L_1 loss as the following formulation. where $\mathcal{N}_\updownarrow$ is the number of live sample in one batch.

$$L_r = \frac{1}{N_l} \sum_{I_i \in live} \|C_i\|_1 \tag{3}$$

Secondly, as an implicit supervision to promote live-live intra-class compactness and live-spoof inter-class separibility, the Triplet Loss is calculated by obtaining a set of feature vectors $\{V\}$ from global average pooling of feature maps from layer E5 to D4 then applying the metric learning-based loss according to the following formula;

$$L_t = \frac{1}{T} \sum_{i=1}^{T} max(d(a_1, pi) - d(a_i, n_i) + m, 0)) \tag{4}$$

$$d(i,j) = \left\| \frac{v_i}{\|v_i\|_2} - \frac{v_j}{\|v_j\|_2} \right\|_2 \tag{5}$$

where $\{a_i, p_i, n_i\}$ denotes the anchor (live), positive (live), and negative (spoof) samples within the ith triplet and T is the number of triplets, $d(i,j)$ represents the euclidean distance between two L2-normalized feature vectors output by the GAP layer, and m is the pre-defined margin constant. The auxiliary classifier is added to the residual learning framework to serve as a spoof cue amplifier and helps the model learn more discriminative spoof cues. The generated spoof cue maps C are added to the input image I to form overlayed images called S which is fed to the auxiliary classifier. While N is the number of samples, z_i is the binary label and q_i is the network predicted label, the auxiliary classifier loss can be calculated as the following.

$$L_a = \frac{1}{N} \sum_{i=1}^{N} z_i log q_i + (1 - z_i) log(1 - q_i) \tag{6}$$

Thus, the loss functions for training this model consisted of the pixel-wise spoof cue regression loss L_r, triplet loss L_t and the auxiliary classification loss L_a which are integrated to establish the total loss L during training. $\alpha_1, \alpha_2, \alpha_3$ are the weights we can set for each of the loss components.

$$L = \alpha_1 L_r + \alpha_2 \sum_{k \in \{E5-D4\}} L_t^k + \alpha_3 La \tag{7}$$

At train and validation phase we used the classifier's output to evaluate the model while in test phase the distance between $\phi(t;\mathcal{W})$ and the center of the live sphere is used to calculate the spoof score of sample t

$$score = \overline{\|\phi(t;\mathcal{W} - c)\|} \tag{8}$$

B. Multi-Task Cascaded Convolutional Neural Networks (MTCNN)

Face detection and alignment are essential to many face applications, such as face recognition and facial expression analysis. MTCNN is a framework where face detection and alignment are integrated using unified cascaded CNNs by multi-task learning [3]. These CNNs consist of three stages where the first stage quickly produces candidate windows through shallow CNN (Proposal Network) to obtain their bounding box regression and employ non-maximum suppression (NMS) to merge highly overlapped candidates. At the second stage, the candidate windows are refined using Refined Net which rejects a large number of non-faces windows. At this stage, the calibration with bounding box regression is performed and the highly overlapped candidates are merged by NMS. Finally, stage 3 uses a more powerful CNN to refine the result and output facial landmarks position. In stage 3, multiple CNNs of 3x3 filters have been designed for face detection. During the training, three tasks to train the detectors were

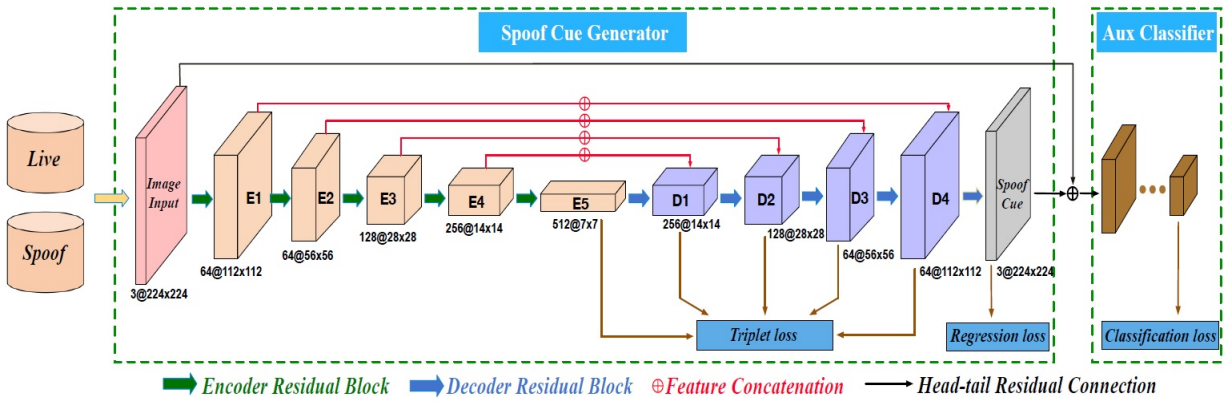

Fig. 1. Residual Learning Network Architecture

leveraged with the loss function consisting of face/non-fact classification, bounding box regression and facial landmark localization loss function. The face/non-fact classification loss function is as follows:

$$L_i^{det} = -(y_i^{det} log(p_i) + (1 - y_i^{det})(1 - log(p_i))) \quad (9)$$

,where p_i is the probability produced by the network that indicates a sample being a face. The notation of $y_i^{det} \in 0, 1$ denotes the ground-truth label.

The bounding box regression loss function is as follows:

$$L_i^{box} = \| \widehat{y}_i^{box} - y_i^{box} \|_2^2 \quad (10)$$

, where y_i^{box} regression target obtained from the network and y_i^{box} is the ground-truth coordinate. There are four coordinates, including left top, height and width, and this $y_i^{box} \in R^4$.

The facial landmark localization loss function is as follows:

$$L_i^{landmark} = \| \widehat{y}_i^{landmark} - y_i^{landmark} \|_2^2 \quad (11)$$

, where $y_i^{landmark}$is the facial landmark's coordinate obtained from the network and $y_i^{landmark}$ is the ground-truth coordinate. There are five facial landmarks, including left eye, right eye, nose, left mouth corner, and right mouth corner, and this $y_i^{landmark} \in R^{10}$.

III. Methodology

A. Dataset

In order to achieved a more generalized FAS system, many have attempted to develop databases that cover different face variations, environments, imaging device and techniques and spoof methods. In this study, the models were trained on our custom combined datasets which included the following 4 FAS datasets (See Table 1).

1) Casia-FASD [4]: contains 50 subjects with 3 fake attacks; warped photo attack, cut photo attack and video attack, all of which were recorded with low, normal and high imaging quality resulting in 12 videos per subject and a total of 600 videos clip.

2) OULU-NPU [5]: consists of 5940 videos of 55 subjects recorded in three different environments via six different smartphones. High quality print and video attacks were created using two different printers and two different display devices.

3) Casia-SURF [6]: dramatically expanded the scale of FAS dataset by recording the data from 1000 subjects with 21000 videos involving 6 print attack types, furthermore, each sample had 3 modalities; RGB, depth and IR.

4) SiW-M [7]: focused on developing a database with diversed types of spoof attacks. Thirteen types of spoof attacks including prints, replays, 3D masks, makeup and partial attacks were implemented in 493 subjects resulting in 1630 videos.

Screenshots from the videos in the datasets were used as inputs of our models. The subsets in each data set were distributed into 3 sets (training, validation and test set) for the training of our model. The subset and number of samples in each of our datasets are shown in Table 2.

B. Data Augmentation

The training, validation and test data were face cropped with MTCNN before they were fed into the model. In the training set, the images were rotated between [-30,30] degrees, selected randomly from the uniform distribution then resized to size 224 x 224 then center cropped. The pixel values were normalized by dividing by 255, subtracting with the mean per channel and dividing by std per channel. For validation and test set, the images were resized and centered cropped to size 224 x 224 then normalized.

C. Models

1) the reformulated FAS : Batch size of 32 was used in a 10-epoch training where the numbers of live and spoof samples in each batch were balanced. $\alpha_1, \alpha_2, \alpha_3$ were set at 5.0, 1.0 and 5.0, respectively. Adam optimizer was chosen with learning rate of 0.0005 and the learning rate that decayed by 0.3 at epoch 5 and 8. The training process was monitored through TensorBoard.

Dataset				Face Variations					Spoof Attacks types					
Name	Year	#Subjects	#Videos	pose	expression	lighting	Camera	Modal types	replay	print	3D mask	makeup	partial	Total num of spoof types
Casia-FASD	2012	50	600	Frontal	No	No	VIS	RGB	1	2	0	0	0	3
OULU-NPU	2017	55	5940	Frontal	No	No	VIS	RGB	1	1	0	0	0	2
Casia-SURF	2018	1000	21000	[-30,30]	No	No	RealSense	RGB/Depth/IR	0	6	0	0	0	6
SiW-M	2019	493	1630	[-90,90]	Yes	Yes	VIS	RGB	1	1	5	3	3	13

TABLE I
DETAILS ABOUT CASIA-FASD, OULU-NPU, CASIA-SURF AND SIW-M DATASET, THEIR FACE VARIATIONS AND SPOOF ATTACK TYPES

Dataset	Train		Validation		Test	
	Set	#Sample	Set	#Sample	Set	#Sample
CASIA-FASD	train	22451	test	32764	-	-
OULU-NPU	train	7288	dev	2699	test	360
CASIA-SURF	train	30831	val	4260	test	1053
SiW-M	train	14633	val	4804	test	28855
Total		75203		44527		44527

TABLE II
THE SUBSET AND NUMBERS OF SAMPLE FROM THE ORIGINAL DATASET THAT WERE INCLUDED IN EACH SET OF OUR COMBINED DATASET.

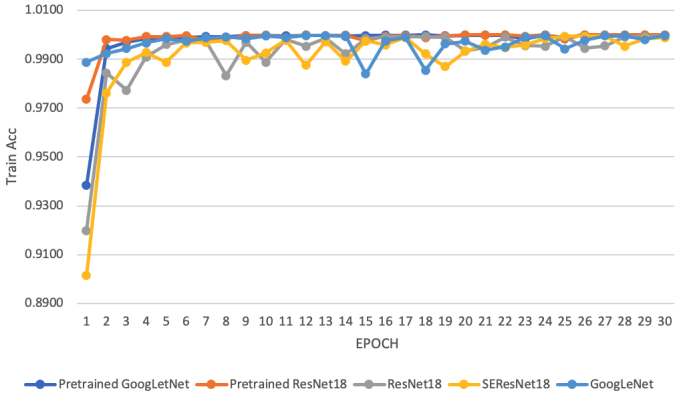

Fig. 2. Train Accuracies of Standard Models

D. Metrics

The following are the matrices used to evaluate our reformulated FAS model.

1) Attack Presentation Classification Error Rate (APCER):

$$APCER = \frac{FP}{(TN/FP)}$$

2) Normal Presentation Classification Error Rate (NPCER):

$$BPCER = \frac{FN}{(FN/TP)}$$

3) Average Classification Error Rate (ACER):

$$ACER = \frac{APCER + NPCER}{2}$$

4) Area Under the Receiver Operating Characteristic Curve (ROC AUC): Area under the curve that is plotted by False Positive Rate (FPR) against True Positive Rate (TPR)

E. Automate Cropping and Live Detection

OpenCV is adopted to detect images in which they will be face-cropped by MTCNN before being fed into the model. The MTCNN does the face-cropping by detecting all bounding boxes of the face and crop them. After the cropped images are detected, images are resize to the size of 224 x 224 then are center cropped. The images are fed into model. The model returns a set of cues. A cue is used to compute a score by calculating a mean and being subtracted from one. A threshold is set to compare a score. If the computed score is higher than the set threshold, the image is real. On the other hand, if the computed score is less than the set threshold, the image is fake. The face images from MTCNN were drawn with OpenCV and result of model showed on top of face image.

IV. RESULTS

As for the results, we compared classification results of the reformulated FAS framework to the well-known GoogLeNet, Pretrained-GoogLeNet, ResNet18, Pretrained ResNet18 and SEResNet18 models on precropped images. The results of GoogLeNet, Pretrained-GoogLeNet, ResNet18, Pretrained-ResNet18, SEResNet18 and reformulated FAS can be seen in the Appendix Table 5, Table 6, Table 7, Table 8, Table 9 and Table 10 respectively. Despite the convergence of losses of the aforementioned models (See Figure 3), the test accuracies of the pretrained models were expectedly low (See Table 4). While GoogLeNet, ResNet18 and SEResNet18 achieved 67.30%, 77.92% and 81.72%, respectively, the Pretrained-GoogLeNet and Pretrained-ResNet18 achieved only 49.77% and 51.60%.

The reformulated FAS achieved 70.12% on our custom datasets. We further analyzed the results by delving deeper into each dataset and noticed that in general our model performed relatively well across all datasets except Casia-SURF (See Table 3). The model only achieved 62.7% on Casia-SURF where we suspect that it could be due to how the images were originally cropped where only the face were present with complete black background.

Therefore, as for the our real-time faceliveness detection system, we further explored GoogLeNet, ResNet18 and SEResNet18. Once integrated the automate cropping, MTCNN and the OpenCV to enable real-time detection to our framework, the reformulated FAS framework outperformed the rest with the empirically found best threshold at 0.96.

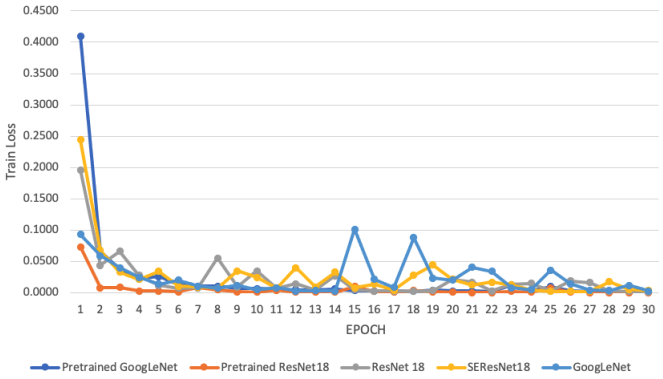

Fig. 3. Train Losses of Standard Models

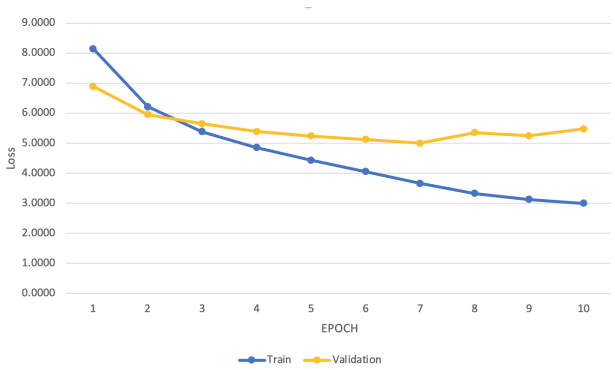

Fig. 4. Train and Validation Loss of FAS Model.

Dataset	Loss	Accuracy	ACER	APCER	NPCER	ROC
OULU-NPU	4.0622	0.9250	0.0783	0.0995	0.0572	0.9568
Casia-SURF	16.6107	0.6273	0.4180	0.4665	0.3695	0.5545
SiW-M	3.6814	0.9555	0.0462	0.0375	0.0550	0.9793
Combined	13.6878	0.7013	0.3342	0.3700	0.2983	0.6504

TABLE III
LOSS AND ACCURACY OF EACH DATASET IN TEST SETS OF THE REFORMULATED FAS.

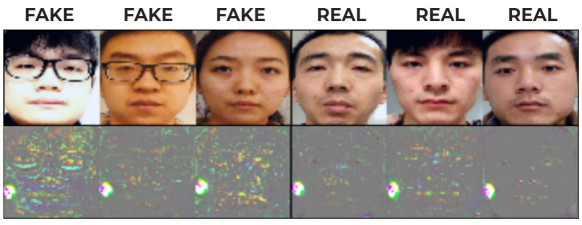

Fig. 5. Example of generated spoof cues from OULU-NPU test set

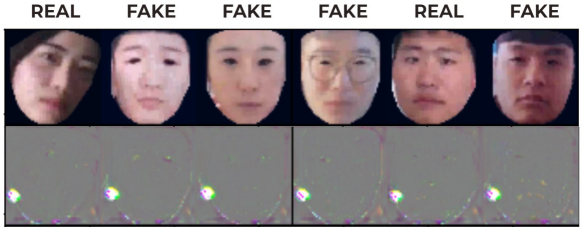

Fig. 6. Example of generated spoof cues from CASIA-SURF test set

Fig. 7. Example of generated spoof cues from SiW-M test set

Models	Test Acc
GoogLeNet	67.30%
Pretrained GoogLeNet	49.77%
ResNet18	77.92%
Pretrained ResNet18	51.60%
SEResNet18	81.72%
Reformulated FAS	70.12%

TABLE IV
TEST ACCURACY OF MODELS.

V. CONCLUSION

In summary, this paper investigates the robustness of the reformulated FAS framework. We also compared the reformulated FAS classification results on our custom datasets with GoogLeNet, Pretrained GoogLeNet, ResNet18, Pretrained-ResNet18 and SEResNet18. We found that non-pretrained models outperformed the pretrained models. the reformulated FAS achieved 70.12% on classification problem on our custom datasets which is unexpectedly low when compared to ResNet18 with 77.92% and SEResNet18 with 81.72%. To further develop the real-time faceliveness detection system, MTCNN and OpenCV were integrated to our models. The integration was done on all 4 models, namely the reformulated FAS, GoogLeNet, ResNet18 and SEResNet18. By integrating MTCNN and OpenCV, it allows real-time detection ability.

Interestingly, once MTCNN and OpenCV were integrated and the models were tested, the reformulated FAS model outperformed other models.

VI. APPENDIX

Epoch	Train Loss	Train Acc	Val Loss	Val Acc
0	0.0929	0.9888	1.7058	0.6768
1	0.0590	0.9923	0.9460	0.7652
2	0.0398	0.9944	1.6899	0.7711
3	0.0248	0.9967	0.6575	0.8774
4	0.0136	0.9985	0.7626	0.8534
5	0.0199	0.9974	0.6521	0.8987
6	0.0104	0.9987	0.8571	0.8852
7	0.0074	0.9991	1.1626	0.8354
8	0.0121	0.9984	0.5080	0.9053
9	0.0049	0.9996	0.7660	0.8822
10	0.0080	0.9989	0.7271	0.8823
11	0.0030	0.9997	0.6691	0.8968
12	0.0036	0.9997	0.8553	0.8847
13	0.0029	0.9996	3.3673	0.7366
14	0.1013	0.9840	0.6342	0.8893
15	0.0214	0.9975	0.7280	0.8688
16	0.0087	0.9990	4.0156	0.5899
17	0.0882	0.9855	1.2735	0.7522
18	0.0236	0.9963	1.3284	0.7288
19	0.0201	0.9974	3.1757	0.7288
20	0.0407	0.9937	1.3416	0.7978
21	0.0344	0.9950	0.8904	0.8766
22	0.0077	0.9990	0.9713	0.8781
23	0.0045	0.9995	3.6045	0.7022
24	0.0359	0.9942	3.6045	0.7022
25	0.0145	0.9978	0.8432	0.8823
26	0.0043	0.9995	0.6686	0.8914
27	0.0036	0.9996	0.9565	0.8786
28	0.0118	0.9981	0.6750	0.8876
29	0.0031	0.9997	1.9061	0.7930

TABLE V
LOSS AND ACCURACY OF TRAIN AND VALIDATION SETS OF GOOGLENET.

Epoch	Train Loss	Train Acc	Val Loss	Val Acc
0	0.4100	0.9383	0.5244	0.8804
1	0.0661	0.9944	0.6845	0.8589
2	0.0383	0.9970	0.5901	0.8899
3	0.0222	0.9984	0.6780	0.8675
4	0.0259	0.9989	0.6290	0.8951
5	0.0138	0.9989	0.6290	0.8951
6	0.0103	0.9993	0.6664	0.8844
7	0.0104	0.9992	0.6078	0.8975
8	0.0069	0.9997	0.5776	0.8985
9	0.0064	0.9997	0.6335	0.8994
10	0.0059	0.9996	0.5968	0.8994
11	0.0045	0.9998	0.5905	0.9005
12	0.0044	0.9997	0.6459	0.9001
13	0.0061	0.9994	0.7086	0.8977
14	0.0038	0.9997	0.7352	0.8953
15	0.0031	0.9998	0.7101	0.8991
16	0.0029	0.9998	0.6429	0.9004
17	0.0025	0.9999	0.6646	0.8961
18	0.0044	0.9995	0.6340	0.9012
19	0.0028	0.9998	0.6595	0.9006
20	0.0031	0.9998	0.6458	0.9014
21	0.0025	0.9998	0.9857	0.8222
22	0.0122	0.9984	0.7380	0.8950
23	0.0032	0.9998	0.9153	0.8657
24	0.0098	0.9990	0.7250	0.9000
25	0.0027	0.9999	0.6775	0.8993
26	0.0024	0.9999	0.6863	0.9018
27	0.0021	0.9999	0.7703	0.8969
28	0.0020	0.9999	0.6526	0.9030
29	0.0017	0.9999	0.6911	0.9027

TABLE VI
LOSS AND ACCURACY OF TRAIN AND VALIDATION SETS OF PRETRAINED GOOGLENET.

Epoch	Train Loss	Train Acc	Val Loss	Val Acc
0	0.0734	0.9736	0.5188	0.8867
1	0.0077	0.9981	0.7238	0.8607
2	0.0084	0.9977	0.6236	0.8926
3	0.0030	0.9993	0.6637	0.8933
4	0.0026	0.9993	0.7098	0.8912
5	0.0014	0.9997	0.7418	0.8732
6	0.0077	0.9976	0.9751	0.8424
7	0.0046	0.9986	0.8022	0.8949
8	0.0015	0.9996	0.7292	0.8994
9	0.0009	0.9998	0.8092	0.8933
10	0.0039	0.9988	0.7563	0.8935
11	0.0009	0.9998	0.8274	0.8938
12	0.0009	0.9997	0.7675	0.8966
13	0.0007	0.9998	1.1590	0.8320
14	0.0096	0.9975	0.7354	0.8981
15	0.0020	0.9994	0.7421	0.8972
16	0.0011	0.9997	0.7520	0.8948
17	0.0035	0.9988	0.8278	0.8923
18	0.0013	0.9996	0.8259	0.8993
19	0.0006	0.9999	0.7989	0.8995
20	0.0004	0.9999	0.8258	0.8995
21	0.0004	0.9999	0.7850	0.8974
22	0.0020	0.9994	0.8339	0.8998
23	0.0006	0.9999	0.9458	0.8737
24	0.0078	0.9983	0.7233	0.9066
25	0.0007	0.9999	0.7922	0.9044
26	0.0005	0.9999	0.7542	0.9028
27	0.0004	0.9999	0.7922	0.9044
28	0.0002	1.0000	0.7956	0.9038
29	0.0005	1.0000	0.7885	0.8989

TABLE VII
LOSS AND ACCURACY OF TRAIN AND VALIDATION SETS OF PRETRAINED-RESNET18.

Epoch	Train Loss	Train Acc	Val Loss	Val Acc
0	0.1961	0.9199	0.6277	0.8327
1	0.0437	0.9844	2.8256	0.7761
2	0.0659	0.9773	1.2124	0.8203
3	0.0277	0.9910	0.4817	0.9044
4	0.0118	0.9960	0.8888	0.8646
5	0.0070	0.9977	0.7510	0.8876
6	0.0080	0.9974	3.1491	0.6275
7	0.0555	0.9832	0.6602	0.8781
8	0.0095	0.9969	1.8312	0.7781
9	0.0347	0.9887	0.4053	0.9140
10	0.0070	0.9980	0.6714	0.8690
11	0.0143	0.9953	0.6427	0.8476
12	0.0042	0.9987	1.0568	0.8476
13	0.0269	0.9921	0.6530	0.8873
14	0.0050	0.9984	0.6757	0.8814
15	0.0031	0.9991	0.6417	0.8890
16	0.0029	0.9992	0.6907	0.8928
17	0.0022	0.9993	0.9115	0.8713
18	0.0037	0.9990	0.8042	0.8463
19	0.0216	0.9934	0.7545	0.8588
20	0.0172	0.9949	0.6770	0.8879
21	0.0031	0.9990	1.4393	0.8239
22	0.0132	0.9956	1.4902	0.8183
23	0.0148	0.9953	0.6235	0.8967
24	0.0028	0.9992	1.0992	0.8304
25	0.0186	0.9945	0.4540	0.9019
26	0.0160	0.9955	0.5700	0.8914
27	0.0031	0.9991	0.7050	0.8854
28	0.0022	0.9994	0.6503	0.8943
29	0.0018	0.9995	0.6764	0.8998

TABLE VIII
LOSS AND ACCURACY OF TRAIN AND VALIDATION SETS OF RESNET18.

Epoch	Train Loss	Train Acc	Val Loss	Val Acc
0	0.2443	0.9014	1.0318	0.7444
1	0.0675	0.9761	0.9889	0.8101
2	0.0329	0.9888	0.4125	0.8943
3	0.0217	0.9927	3.0989	0.7012
4	0.0339	0.9887	0.5454	0.8906
5	0.0108	0.9966	0.6104	0.8759
6	0.0094	0.9970	0.4025	0.9164
7	0.0076	0.9976	1.5695	0.7878
8	0.0348	0.9896	2.1504	0.7738
9	0.0246	0.9926	0.4945	0.9024
10	0.0072	0.9978	0.6618	0.8564
11	0.0399	0.9875	0.5214	0.8930
12	0.0090	0.9972	3.3848	0.7554
13	0.0332	0.9891	0.5180	0.8880
14	0.0081	0.9975	1.1378	0.8580
15	0.0135	0.9959	0.7043	0.8867
16	0.0042	0.9988	3.0765	0.7184
17	0.0276	0.9921	1.0860	0.8079
18	0.0443	0.9870	1.5652	0.8044
19	0.0214	0.9932	0.5424	0.8869
20	0.0127	0.9961	1.1066	0.8482
21	0.0171	0.9947	0.8272	0.8577
22	0.0125	0.9958	0.4554	0.9131
23	0.0050	0.9985	0.5048	0.9013
24	0.0024	0.9994	0.4822	0.9092
25	0.0022	0.9994	0.4997	0.9062
26	0.0019	0.9995	1.2038	0.8269
27	0.0175	0.9953	0.7917	0.8763
28	0.0054	0.9985	0.3879	0.9275
29	0.0037	0.9989	0.6420	0.8979

TABLE IX
LOSS AND ACCURACY OF TRAIN AND VALIDATION SETS OF SERESNET18.

	Train	Val				
Epoch	Loss	Loss	Accuracy	ACER	APCER	NPCER
1	8.1426	6.8891	0.6996	0.3253	0.3609	0.2897
2	6.2186	5.9502	0.7467	0.2707	0.3131	0.2284
3	5.3801	5.6473	0.7867	0.2245	0.2848	0.1641
4	4.8507	5.3902	0.8331	0.1721	0.2374	0.1069
5	4.4299	5.2393	0.8621	0.1418	0.1997	0.0837
6	4.0551	5.1226	0.8745	0.1293	0.1825	0.0762
7	3.6593	5.0033	0.8885	0.1141	0.1599	0.0683
8	3.3275	5.3518	0.8865	0.1159	0.1577	0.0740
9	3.1259	5.2448	0.8939	0.1085	0.1547	0.0624
10	2.9995	5.4757	0.8932	0.1097	0.1503	0.0692

TABLE X
LOSS AND ACCURACY OF TRAIN AND VALIDATION SETS OF THE REFORMULATED FAS.

REFERENCES

[1] H. Feng, Z. Hong, H. Yue, Y. Chen, K. Wang, J. Han, J. Liu, and E. Ding, "Learning generalized spoof cues for face anti-spoofing," *arXiv preprint arXiv:2005.03922*, 2020.

[2] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[3] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, 2016.

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

[5] Z. Boulkenafet, J. Komulainen, L. Li, X. Feng, and A. Hadid, "Oulu-npu: A mobile face presentation attack database with real-world variations," in *2017 12th IEEE international conference on automatic face & gesture recognition (FG 2017)*. IEEE, 2017, pp. 612–618.

[6] S. Zhang, A. Liu, J. Wan, Y. Liang, G. Guo, S. Escalera, H. J. Escalante, and S. Z. Li, "Casia-surf: A large-scale multi-modal benchmark for face anti-spoofing," *IEEE Transactions on Biometrics, Behavior, and Identity Science*, vol. 2, no. 2, pp. 182–193, 2020.

[7] Y. Liu, J. Stehouwer, A. Jourabloo, and X. Liu, "Deep tree learning for zero-shot face anti-spoofing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4680–4689.