# LiveFace: A Multi-Task CNN for Fast Face-Authentication

Xiaowen Ying, Xin Li, Mooi Choo Chuah
*Department of Computer Science and Engineering, Lehigh University*
xiy517@lehigh.edu, xil915@lehigh.edu, chuah@cse.lehigh.edu

***Abstract*—Modern face recognition systems are accurate but they are vulnerable to different types of spoofing attacks. To solve this problem, conventional face authentication systems typically employ an additional module to analyze the liveness of the input faces before feeding it into the face recognition module. Such two-stage designs not only suffer from longer processing time but also require more storage and resources, which are usually limited on mobile and embedded platforms.**

**In this paper, we propose a multi-task Convolutional Neural Network(CNN), namely LiveFace, for face-authentication. Given an input face image, LiveFace generates two outputs through a single stage: (i) a face representation that can be used for identification or verification, and (ii) the corresponding liveness score. The two tasks share lower layers to reduce the computation cost. Experimental results using three datasets show that our model achieves a comparable performance on both face recognition and anti-spoofing tasks but much faster than conventional authentication systems. In addition, we have implemented a prototype of our scheme on Android phones and demonstrated that our scheme can run in real-time on three Android devices that we have tested.**

***Index Terms*—Face Recognition, Face Anti-spoofing, Liveness Detection**

## I. Introduction

Modern deep-learning based face recognition systems have achieved comparable performances to human-beings [1] [2], and the accuracy continues to improve by using more training data and deeper neural networks [3] [4]. However, these systems are vulnerable to face-spoofing attacks. For example, we can easily fool the system by showing a photo of a valid user in front of the camera. In addition, the size and the execution speed of the model also become crucial concerns for mobile and embedded face authentication systems due to their limited storage and computational resources.

While face recognition has been intensely studied for decades, face anti-spoofing is still a relatively new topic. Different approaches have been explored to defend face spoofing attacks, including both traditional methods which primarily use hand-crafted features, and CNN-based methods which utilize deep neural networks for feature extraction. Existing approaches focus on solving face recognition and face anti-spoofing independently, and to the best of our knowledge, none of them combines these two tasks into a single pass.

Inspired by the success of Multi-Task Learning(MTL) on many face-related tasks [5] [6] [7], we propose LiveFace, a multi-task CNN that solves these two crucial problems through a single stage to speed up the authentication procedure. We believe that there are some common features that can be shared between the two tasks. Our goal is to minimize the additional cost of adding a second task by sharing some layers between the two tasks. Figure 1 shows the difference between a typical face authentication system and the LiveFace.

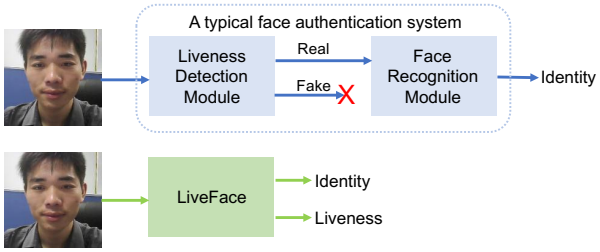


Fig. 1: A comparison between typical face authentication systems and our scheme. Top row shows the procedure of a typical face authentication system, while the bottom row shows the procedure of our scheme.

The performance of our model is evaluated on popular face recognition and face anti-spoofing datasets including LFW Benchmark [8], CASIA-FASD [9] and Replay-Attack Database [10]. A surprising finding is that multi-task learning also improves the performance of the face anti-spoofing task compared to training it independently. Execution speed evaluation shows that our multi-task model runs as fast as the single-task models and much faster than any combination of existing methods. Details of our experiments and evaluations are discussed in Section IV.

Our main contributions can be summarized as follows:

- We propose a multi-task CNN that solves face recognition and face anti-spoofing via a single model to optimize the authentication procedure.
- We propose a training strategy to train our multi-task model from two different datasets.
- Our model achieves comparable performance on both tasks compared to other state-of-the-art methods but runs much faster and requires fewer resource.
- We have also implemented our scheme on Android phones and evaluated their execution time.

## II. Related Work

### A. Face Recognition

The goal of a face recognition system is to identify a person from a digital image. Traditional face recognition algorithms extract facial features from handcrafted local visual descriptors such as LBP [11], SIFT [12], HOG [13], etc. According to [14], these handcrafted feature extractors suffer from two drawbacks. First, manually designing an optimal encoding method is difficult. Second, handcrafted descriptors usually lead to a less informative and less compact representation.

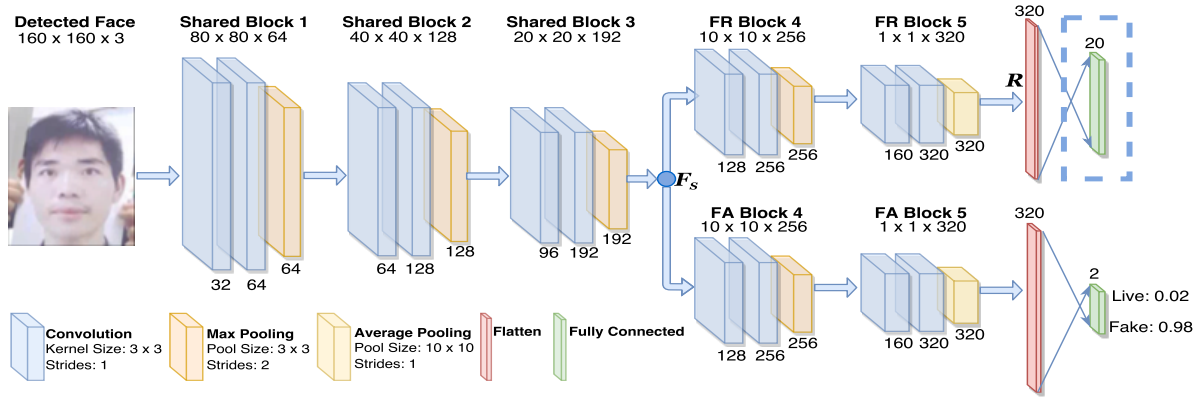


Fig. 2: The model architecture of LiveFace. The first three blocks are shared between two tasks, then the model is split into two branches. The number above each block is the output shape of the pooling layer. The output dimension of the Face Anti-spoofing (FA) branch is two, which corresponds to the probability of "live" and "fake". During training, the last layer of the Face Recognition (FR) branch is a Softmax FC layer that has the size equals to the number of people in the dataset. Once the training is done, this layer is removed and the output of the FR branch is the learned representation $R$. As illustrated in the legend, same type of layers use the same configuration such as kernel size and strides.

Modern CNN-based face recognition systems have achieved the same level of accuracy as human-beings.Taigman et al. [1] proposes to perform 3D face alignment before fedding it into a deep neural network. Their work was later extended by a series of DeepID related papers [15] [2] which explore several new ideas to improve the performance by stacking a large number of CNNs, which results in a complicated final model. Schroff et al. [4] present FaceNet that achieves better performance than DeepFace using a massive dataset and a novel loss function. Parkhi et al. [3] propose to train a very deep neural network for face recognition without any embellishments and achieve results comparable to the state-of-the-art.

### B. Face Anti-spoofing

The goal of face anti-spoofing, also known as liveness detection, is to determine if an input face image is genuine or fake. In this paper, we assume that we can only obtain the input face through a built-in camera, thus other methods that utilize input data from special sensors such as IR sensor or depth-camera will not be discussed.

**Traditional Face Anti-spoofing Methods.** Traditional methods focus on analyzing the texture or the quality of the input face images. Patel et al. [16] propose to analyze the Moiré pattern of the given face images and find it very effective against replay video attacks. Agarwal et al. [17] achieves a good performance by extracting Haralick features after Redundant Discrete Wavelet Transform (RDWT). However, their performance is sensitive to the number of frames. There are also a few works that utilize motion clues such as eye-blinks [18] and lips movements [19]. However, such approaches will fail when attackers replay recorded videos.

A significant drawback of traditional face anti-spoofing methods is that hand-crafted features may not be generalizable, hence the performance of these approaches usually fluctuate a lot on different datasets.

**Deep Learning-based Face Anti-spoofing Methods.** Li et al. [20] apply an SVM to the convolutional responses of a CNN for anti-spoofing. Yang et al. [21] propose to use a stack of images with different scales cropped from the original frame as the input of a CNN model. Xu et al. [22] proposes to leverage the temporal features between frames via a LSTM-CNN networks. Atoum et al. [23] proposes to use a Fully Convolutional Neural Network to estimate the depth of the input face, the generated depth map is then fed into an SVM to distinguish between live and spoofed faces.

## III. Proposed Method

### A. Model Architecture

Our model structure is a modified version of CASIA-Net [24]. Compared to its original structure, several modifications have been applied. First, we change the input shape of the model from 100x100x1 to 160x160x3 to include more information. Second, Batch Normalization layers are included to accelerate the convergence. Third, several convolution blocks are added as a new branch for the face anti-spoofing task.

Figure 2 describes the architecture of our proposed network. The first $K$ ($K = 3$ in Figure 2) blocks in the proposed model are shared between the two tasks. Each convolutional layer is followed by a Batch Normalization operation and a ReLU [25] activation except the last one in both branches. In addition, a dropout layer is applied after the Flatten layers in both branches.

During our multi-task learning, the shared blocks learn an intermediate feature map $F_s$ for both face recognition and face anti-spoofing tasks. The learned feature map $F_s$ may include some common features $F_c$ useful to both branches and task-specific features $F_{fr}$ and $F_{fa}$ that can be later decoupled by subsequent task-specific blocks. Based on the shared feature map $F_s$, each task-specific branch further learns the final representation for its classification task. Eventually, the final

representation $R$ can be fed into the K-way Softmax classifier and the probability assigned to the $j$th class is:

$$P(y=j|R) = \frac{exp(R^T W_j + b_j)}{\sum_{k=1}^{K} exp(R^T W_k + b_k)}, \quad (1)$$

where $W$ and $b$ are the weights and bias matrices of the FC layer. The output of the Softmax classifier is a probability distribution, and the predicted class is defined as the one with the highest probability.

### B. Data Preprocessing

While some existing approaches involve domain-specific knowledge into the preprocessing step [26] [27] [23], we train our model to learn all features by itself through end-to-end training. A face detector [6] is applied to each input image, and the detected faces are then aligned and cropped based on the position of the eyes. The input images are normalized to zero-center and finally resized to 160x160 in RGB color space before being fed into the model.

### C. Training Strategy

Training a multi-task model typically requires the training data to include multiple labels. However, currently there is no public dataset that can be used to train FR & FA tasks simultaneously. On the one hand, large face datasets with a great variety of gender, age, and race are available to train a robust face recognition model but they do not include spoofing samples. On the other hand, only a small number of identities are included in the popular anti-spoofing datasets, and such datasets lack varieties.

To solve this problem, we propose to use a two-step training strategy to train our model using two datasets.

**Step 1:** Considering that a robust face representation is more difficult to learn, we first remove the FA branch and train the FR branch independently from scratch using a large human face dataset.

**Step 2:** Based on the pre-trained model from step 1, we manually add the identity label to the training samples in the anti-spoofing dataset and use it to do the multi-task training. In this step, the weights in FR Blocks 4 & 5 are frozen.

The cross-entropy loss function is employed in both branches. Given the ground truth vector $y$ and the prediction vector $\hat{y}$, the cross-entropy loss is calculated by:

$$L(y, \hat{y}) = -\frac{1}{K}(\sum_{k=1}^{K} y_k \cdot log(\hat{y}_k) \quad (2)$$

where K is the number of face identities.

During the multi-task training in Step 2, the goal is to minimize the combined loss:

$$L_{total} = \alpha L_{fr} + \beta L_{fa} \quad (3)$$

where $\alpha$ and $\beta$ are factors that control the importance of each task. We define them as $\alpha, \beta \in [0, 1]$ and $\alpha + \beta = 1$.

While conventional transfer learning approaches typically freeze the base of a pre-trained model and fine-tune its later stage layers, we propose to freeze the later layers and fine-tune the front shared layers. This design is based on the following intuition – let us assume that a non-linear mapping $G_{fr}$ from $F_s$ to the face representation $R$ is learned during Step 1. While training the FA branch during Step 2, the mapping $G_{fr}$ is preserved to prevent identity features $F_{fr}$ from being overwritten by back-propagation from the FA branch. It also serves as a regularization term that forces the FA branch to focus on face-related features and reduces the model's attention on non-generalizable features.

For example, the border of the photos and screens is an obvious clue that distinguishes spoofing attacks, but it is not generalizable because attackers can easily hide the borders by moving them closer to the camera. Without any regularization during training, the CNN is likely to pay too much attention to these non-generalizable features and ignore some key features. The effectiveness of the proposed training strategy is verified by experimental results in the following section.

## IV. Experiments

### A. Datasets

**CASIA-WebFace** [24] is a large-scale human face dataset containing 494,414 images of 10,575 subjects. All the images are collected from the Internet. In this paper, the CASIA-WebFace is only used to train the FR branch in the first step.

**Labeled Faces in the Wild (LFW)** [8] is a popular benchmark for evaluating a face verification algorithm. The dataset contains more than 13,233 images of 5,749 subjects collected from the web, and the test set contains 6000 pairs of faces in 10 splits labeled with "same" or "not same". In this paper, we use LFW to evaluate the performance of the FR branch in our model.

**CASIA-FASD** [9] is a face anti-spoofing dataset containing 600 video clips of 50 subjects. Three different types of face spoofing attacks are implemented, which include warped photo attack, cut photo attack, and video attack. Each subject has 12 videos (3 genuine and 9 fake), and three different type of spoof attacks are included. The entire dataset is split into a training set containing 20 subjects and a testing set containing 30 subjects.

**Replay-Attack** [10] is a face anti-spoofing dataset containing 1,300 videos of photo and video attacks of 50 subjects under different lighting conditions. The whole dataset is divided to a training set with 15 subjects, a development set with 15 subjects and a testing sets with 20 subjects.

### B. Experimental Parameter Settings and Setup

We use the Keras [28] framework to implement the proposed scheme. During training Step 1, the learning rate is initially set to 0.01 and reduces every 5 epochs by a factor of 0.1. During training Step 2, the learning rate is initially set to 0.001 and reduces every 2 epochs with a factor of 0.4. $\alpha$ and $\beta$ which control the relative importance of the two tasks are experimentally determined to be 0.7 and 0.3.

During the whole training process, a SGD optimizer with a momentum of 0.9 is used. The batch size is set to 8. We augment the data using random horizontal flipping.

To evaluate the performance of face recognition branch, we follow the "unconstrained" protocol [8] and evaluate the

TABLE I: Performance comparison between different model structure. FR-ACC is the verification accuracy on LFW for the FR branch, while FA-EER and FA-HTER is the EER and HTER on CASIA-FASD for the FA branch.

| # of Shared Blocks | 0 | 1 | 2 | **3** | 4 | 5 |
|---|---|---|---|---|---|---|
| FR-ACC (%) | 97.08 | 97.01 | 96.97 | **97.02** | 97.00 | 96.83 |
| FA-EER (%) | 2.22 | 2.78 | 2.78 | **0.83** | 2.96 | 8.15 |
| FA-HTER (%) | 1.85 | 2.04 | 2.59 | **0.56** | 1.85 | 6.85 |

performance on LFW benchmark. For each face pair in LFW, we calculate the cosine similarity between the two face representations. The mean accuracy from 10 fold cross-validation is reported. Given two face representations $R_a$ and $R_b$, the cosine similarity is defined as:

$$S = \frac{R_a \cdot R_b}{||R_a||R_b|}, \tag{4}$$

To evaluate the performance of our FA branch, we follow the protocol associated with CASIA-FASD and Replay-Attack. For each of the dataset, the training set is used for training and the Equal Error Rate(EER) and the Half Total Error Rate (HTER) is reported on the testing set. The development set in Replay-Attack is only used to ensure convergence during the training process. Since the dataset is composed of videos, the video-wise classification is based on the average score calculated from all video frames in the test set.

EER and HTER are two common metrics to evaluate a biometric system. EER is the common rate where the False Acceptance Rate (FAR) equals to the False Rejected Rate (FRR). The HTER is defined as:

$$HTER = \frac{FAR + FRR}{2} \tag{5}$$

Because these two metrics are both related to the error rate, the lower the EER and HTER, the better the model is.

### C. Impacts of Model Structure

To study how different model structures impact the final performance, we conducted a series of experiments using different models where we vary the number of shared blocks.

First, for each of the FR & FA tasks, we train a model that contains only the respective task. Each model is trained from scratch by using the corresponding dataset (CASIA-WebFace for the face recognition model and CASIA-FASD for the face anti-spoofing model). The results of these two models correspond to the values reported in the first column in Table I where the number of shared blocks is 0. Then, we start combining the two models into a multi-task model and gradually increase the number of shared blocks. As we can see, the model with 3 shared blocks outperforms others. Thus, it is used as our final model.

### D. Impacts of Training Strategy

To illustrate the advantages of the proposed training strategy, we trained the LiveFace model with three different training strategies. Training Step 1 described in section 3.3 is first completed for all the strategies. Based on the trained model from Step 1, we test the following training strategies during the second training step. The CASIA-FASD dataset is used as an example.

TABLE II: FR Accuracy on LFW of the resulting model using different strategy.

| Strategy | Baseline | Strategy 1 | Strategy 2 | Proposed |
|---|---|---|---|---|
| Accuracy (%) | 97.08 | 97.08 | 96.83 | 97.02 |

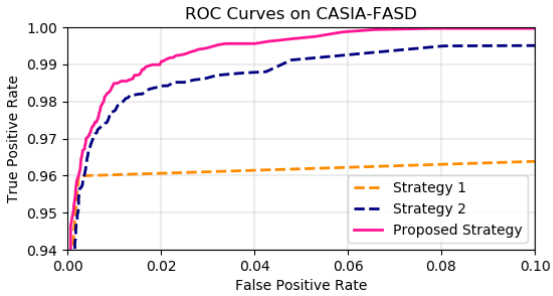


Fig. 3: Frame-based ROC curves of the FA task using different training strategy on CASIA-FASD.

**Strategy 1**: Freeze all the shared blocks and FR blocks, only train the FA blocks.

**Strategy 2**: All blocks are trainable.

**Proposed Strategy**: Freeze only FR blocks, train the FA blocks and the shared blocks.

For **FR** task, the face verification accuracy on LFW benchmark is reported in Table II. The model obtained after Step 1 is used as the baseline. Since Strategy 1 freezes all the FR related blocks, the performance is exactly the same as the baseline model. For **FA** task, the corresponding Receiver Operating Characteristic (ROC) curves of different strategies are compared in Figure 3. The frame-based results are used to plot the curves to show a more discriminative comparison since there are more video frames.

As we can see from Figure 3 and Table II, Strategy 1 keeps the best performance for the FR task, but it heavily hampers the learning of FA task. Strategy 2 allows the FA branch to tune the shared block layers, which increases the performance of FA task; however, it suffers from significant accuracy drop on FR task. Our proposed strategy outperforms others on the FA performance (as shown in Figure 3) and only has trivial impacts (0.06%) on the FR task.

Based on the experimental results of different training strategies, we have the following analysis:

1) For Strategy 1, the weights in the shared blocks and FR blocks are frozen to protect all the features for FR learned from step 1. Only the FA blocks were trained to extract the useful information from the existing intermediate features map $F_s$. However, the current feature map $F_s$ is likely to include only few information that can be leveraged to distinguish between live and spoofed faces.

2) For Strategy 2, since all the weights are adjustable, the learned face representation $R$ is less stable during the training process in step 2. The FR branch still needs to perform the FR task on the anti-spoofing dataset but the adjustments in the weights may affect the learned features in $F_s$ and $R$ from step 1 such that they degrade the FR accuracy.

3) For our proposed strategy, fine-tuning the shared blocks allows some useful information for anti-spoofing to be added

TABLE III: Performance Comparison on LFW Benchmark when # of networks = 1.

| Method | Training Data | | Accuracy |
|---|---|---|---|
| | Images | People | |
| DeepID2-Single [2] | 0.20M | 10,177 | 95.43% |
| DeepFace-Single [1] | 4.40M | 4,030 | 95.92% |
| CASIA-NET [24] | 0.49M | 10,575 | 96.13% |
| VGG-Faces [3] | 2.60M | 2,622 | 98.95% |
| FaceNet [4] | 260.00M | 8,000,000 | 99.63% |
| Our Model | 0.49M | 10,575 | 97.02% |

TABLE IV: Performance Comparison on CASIA-FASD and Replay-Attack In Terms of EER and HTER.

| Method | CASIA-FASD | | Replay-Attack | |
|---|---|---|---|---|
| | EER% | HTER% | EER% | HTER% |
| LBP-TOP [29] | 10.0 | - | 7.9 | 7.6 |
| DPCNN [20] | 4.50 | - | 2.90 | 6.10 |
| LSTM-CNN. [22] | 5.17 | 5.93 | - | - |
| Color-Texture [27] | 6.20 | - | 0.40 | 2.90 |
| Fisher-Vector [30] | 2.80 | - | **0.10** | 2.20 |
| Haralick Features [17] | 1.10 | - | - | - |
| Depth-based CNN [23] | 2.85 | 2.52 | 0.86 | 0.75 |
| Patch+Depth CNN [23] | 2.67 | 2.27 | 0.79 | 0.72 |
| Our Model | **0.83** | **0.56** | 0.42 | **0.13** |

to the shared feature map $F_s$. Freezing the FR blocks allows the mapping $G_{fr}$ to be preserved for regularizing the resulting $F_s$ to keep useful information for face recognition.

### E. Performance Comparison

We compare the performance of our model with other state-of-the-art methods. For the face recognition task, the face verification accuracy on LFW is used for comparison. While many existing works use an ensemble of several networks to achieve higher performances, we only list the results of their single-net version for fair comparison. For the face anti-spoofing task, the results on CASIA-FASD and Replay-Attack datasets are used for comparison.

Table III shows the performance comparison for face recognition task on LFW. As we can see, our model achieves comparable performances to state-of-the-art methods by using limited training data. For example, comparing to VGG-Faces [3], our training data are 81% less than theirs but our accuracy is only 1.93% lower.

Table IV shows the performance comparison for the face anti-spoofing task on CASIA-FASD and Replay-Attack. On the CASIA-FASD dataset, we improve the best EER by 24.55% and HTER by 75.33%. On the Replay-Attack dataset, we improve the best HTER by 82.64%.

It is worth mentioning that our model works effectively on both datasets, while previous works typically leave a huge gap between the results on different datasets. For example, Fisher-Vector [30] achieves 0.10% EER on Replay-Attack but only has 2.20% HTER on the same dataset and 2.80% EER on CASIA-FASD. This means our model works better on learning those truly discriminative and generalizable features.

### F. Execution Speed Comparison

To show the advantages of the running speed of our method, we compare the execution speed of our method with other methods. For fair comparison, we carefully follow the papers and replicate the structures of these methods to test the execution speed on the same environment. We use a laptop that has 1.4GHz Intel Core i5 CPU with 8GB RAM running Mac OS High Sierra. All the preprocessing steps involved in the different methods are excluded in the evaluation.

TABLE V: Execution Speed Comparison (on CPU).

| Method | Task | Speed (ms/frame) |
|---|---|---|
| DeepFace-Single [1] | FR | 429.42 |
| CASIA-NET [24] | FR | 66.84 |
| VGG-Faces [3] | FR | 1064.47 |
| FaceNet (NN1) [4] | FR | 1956.42 |
| Depth-based CNN [23] | FA | 876.75 |
| Patch + Depth CNN [23] | FA | 960.33 |
| Our Model | **FR + FA** | 224.52 |

TABLE VI: Execution Speed Evaluation on Mobile Phones.

| Device | CPU | Cores | Memory | Speed (ms/frame) |
|---|---|---|---|---|
| Samsung Galaxy S5 | Snapdragon 801 | 4 | 2GB | 397.72 |
| Samsung Galaxy S8 | Snapdragon 835 | 8 | 4GB | 253.03 |
| OnePlus 5 | Snapdragon 835 | 8 | 8GB | 238.33 |

For face recognition methods, the execution time is counted from the moment an image is fed until the face representation is obtained. For the face anti-spoofing methods, the execution time is evaluated from feeding in an image until the final liveness score comes out. The final reported time is an average of testing 100 times. All tests are using CPU only, i.e. without GPU acceleration or any speed-up operation that could accelerate CPU processing.

Table V shows the results of the speed evaluation. We can see that our model runs faster than many single-task model. If we combine an existing FR method with a FA method to form a conventional two-step authentication system, our method will be much faster. For example, if one uses Depth-based CNN [23] for FA before using CASIA-NET [24] for FR, the total execution time is 943.59 (876.75 + 66.84) ms/frame while ours is only 224.52 ms/frame. We want to highlight that our proposed scheme is an end-to-end CNN model, thus it runs much faster with a GPU acceleration. It only takes **1.57** ms per frame (FR+FA) on a NVIDIA GTX 1080 Ti GPU.

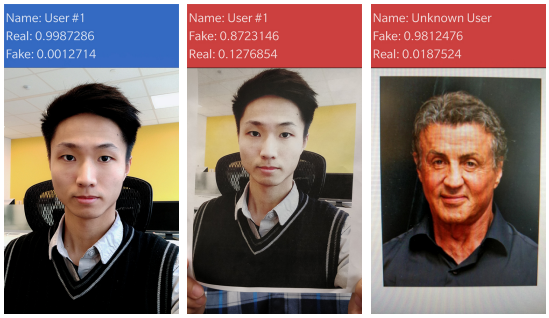


Fig. 4: Screenshots of our mobile App in three different cases. Left: A valid user. Middle: A photo attack. Right: An unknown user.

### G. Mobile Implementation

To test the performance of our model on our targeting application scenario, we implement a prototype of our scheme

on Android phones as shown in Figure 4.

We deploy our application on three different models and evaluate the execution speed in the same way we described in section 4.6. The three devices we tested are (i) Samsung Galaxy S5 which has a 4 Cores Snapdragon 811 CPU and 2 GB memory, (ii) Samsung Galaxy S8 which has a 8 cores Snapdragon 835 CPU and 4 GB memory, and (iii) OnePlus 5 which has a 8 cores Snapdragon 835 CPU and 8GB memory. Our results are reported in Table VI. The results show that having faster CPUs and more memories improve the execution speed of our proposed scheme. We have not optimized our implementation on mobile phones yet so we anticipate that we can achieve better performance with further optimizations.

## V. Conclusion

Face recognition and face anti-spoofing are the two most important procedures in a face authentication system. Unlike existing methods that solve these two tasks independently, this paper introduces a novel method that solves them together through a single stage by using a multi-task Convolutional Neural Network. In our model, lower-level features are shared between two tasks to reduce the computational cost and accelerate the authentication procedure. Since there is no dataset that can be directly used to train these two tasks, we propose a special training strategy to help the multi-task model learn from two datasets. We believe that this training strategy can also be applied to other similar multi-task models.

Performance comparison with other state-of-the-art methods shows that our model achieves comparable results on both tasks. Interestingly, our anti-spoofing branch outperforms existing approaches in three out of four metrics of two datasets. Executing speed comparison shows that our model runs even faster than many single-task model, and runs much faster than any combination of the existing FA & FR methods.

We have also implemented a prototype of our scheme and evaluated it on three types of Android phones. Our current implementation does not use the GPUs available in these phones. Utilizing such GPUs available in Android phones to improve our performance further will be our near future work.

## VI. Acknowledgement

This work is partially supported by a gift from Qualcomm Inc. and a GPU donated by NVIDIA. The author would like to thank Jacob Nelson for his helpful feedback to the manuscript.

## References

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

[2] Y. Sun, Y. Chen, X. Wang, and X. Tang, "Deep learning face representation by joint identification-verification," in *Advances in neural information processing systems*, 2014, pp. 1988–1996.

[3] O. M. Parkhi, A. Vedaldi, A. Zisserman *et al.*, "Deep face recognition." in *BMVC*, vol. 1, no. 3, 2015, p. 6.

[4] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.

[5] C. Zhang and Z. Zhang, "Improving multiview face detection with multi-task deep convolutional neural networks," in *Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on*. IEEE, 2014.

[6] 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.

[7] X. Yin and X. Liu, "Multi-task convolutional neural network for pose-invariant face recognition," *IEEE Transactions on Image Processing*, 2017.

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

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

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

[11] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on pattern analysis and machine intelligence*, 2002.

[12] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, 2004.

[13] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. IEEE, 2005.

[14] Z. Cao, Q. Yin, X. Tang, and J. Sun, "Face recognition with learning-based descriptor," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 2707–2714.

[15] Y. Sun, X. Wang, and X. Tang, "Deep learning face representation from predicting 10,000 classes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1891–1898.

[16] K. Patel, H. Han, A. K. Jain, and G. Ott, "Live face video vs. spoof face video: Use of moiré patterns to detect replay video attacks," in *Biometrics (ICB), 2015 International Conference on*. IEEE, 2015.

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

[18] G. Pan, L. Sun, Z. Wu, and S. Lao, "Eyeblink-based anti-spoofing in face recognition from a generic webcamera," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. IEEE, 2007.

[19] K. Kollreider, H. Fronthaler, M. I. Faraj, and J. Bigun, "Real-time face detection and motion analysis with application in "liveness" assessment," *IEEE Transactions on Information Forensics and Security*, 2007.

[20] L. Li, X. Feng, Z. Boulkenafet, Z. Xia, M. Li, and A. Hadid, "An original face anti-spoofing approach using partial convolutional neural network," in *Image Processing Theory Tools and Applications (IPTA), 2016 6th International Conference on*. IEEE, 2016, pp. 1–6.

[21] J. Yang, Z. Lei, and S. Z. Li, "Learn convolutional neural network for face anti-spoofing," *arXiv preprint arXiv:1408.5601*, 2014.

[22] Z. Xu, S. Li, and W. Deng, "Learning temporal features using lstm-cnn architecture for face anti-spoofing," in *Pattern Recognition (ACPR), 2015 3rd IAPR Asian Conference on*. IEEE, 2015, pp. 141–145.

[23] Y. Atoum, Y. Liu, A. Jourabloo, and X. Liu, "Face anti-spoofing using patch and depth-based cnns," in *Biometrics (IJCB), 2017 IEEE International Joint Conference on*. IEEE, 2017, pp. 319–328.

[24] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Learning face representation from scratch," *arXiv preprint arXiv:1411.7923*, 2014.

[25] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[26] A. da Silva Pinto, H. Pedrini, W. Schwartz, and A. Rocha, "Video-based face spoofing detection through visual rhythm analysis," in *Graphics, Patterns and Images (SIBGRAPI), 2012 25th SIBGRAPI Conference on*. IEEE, 2012, pp. 221–228.

[27] Z. Boulkenafet, J. Komulainen, and A. Hadid, "Face anti-spoofing based on color texture analysis," in *Image Processing (ICIP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 2636–2640.

[28] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[29] T. de Freitas Pereira, J. Komulainen, A. Anjos, J. M. De Martino, A. Hadid, M. Pietikäinen, and S. Marcel, "Face liveness detection using dynamic texture," *EURASIP Journal on Image and Video Processing*, vol. 2014, no. 1, p. 2, 2014.

[30] Z. Boulkenafet, J. Komulainen, and A. Hadid, "Face antispoofing using speeded-up robust features and fisher vector encoding," *IEEE Signal Processing Letters*, vol. 24, no. 2, pp. 141–145, 2017.