An On-Device Face Verification App for Android

Shaheera Uqaili, Pritesh Soni
TU Darmstadt
shaheera.uqaili@stud.tu-darmstadt.de, pritesh.soni@stud.tu-darmstadt.de

Abstract

Face recognition technology is a biometric technology based on the identification of facial features [1]. In this project, we develop a mobile application that performs real-time face verification on the device entirely without relying on cloud-based services. This ensures enhanced security, preserves privacy, reduces latency, and ensures offline functionality—making it suitable for secure mobile authentication, contactless payments, and personalized services. The application supports user registration, captures facial images, and pre-processes them through alignment and cropping using MTCNN. A face template is then extracted using the MobileFaceNet face recognition model, trained via a teacher-student distillation framework. During verification, the captured face is compared with the enrolled template using cosine similarity. If the similarity score exceeds a defined threshold (e.g., 0.3), the system confirms a match; otherwise, it returns a "Face not matched" error. The viability of precise and effective face verification on mobile devices is demonstrated by this small and privacy-oriented system.

1 Introduction

Face detection algorithm is to find out the coordinate system of all faces in one image. This is the process of scanning the entire image to determine whether the candidate area is a face. The output of the face coordinate system can be square, rectangular, etc. The face position is the coordinate position of the face feature in the face detection coordinate system [1]. The deep learning framework basically implements some current good positioning technologies. Compared with face detection, the calculation time of face positioning algorithm is much shorter.

Most existing solutions depend on cloud computing, raising concerns about privacy, data leakage, and connectivity. To address these limitations, we propose an on-device face-verification system for Android platforms that operates without internet access, ensuring that data never leaves the device.

As we develop an on-device application for face verification, template-based enrollment using MTCNN, a neural network which detects faces and facial landmarks on images, and then using MobileFaceNets, an efficient CNN model tailored for high-accuracy real-time face verification on mobile and embedded devices, a deep neural network used for extracting features from an image of a person's face. For comparison of images, we have used a threshold of 0.3 of cosine similarity to verify if the user is enrolled in the application.

Face recognition technology is most used for attendance access control, security, and finance. However, it is also starting to be used in logistics, retail, smartphones, transportation, education, real estate, government management, entertainment advertising, network information security, and other areas. Face recognition can help with early warning of dangerous circumstances and tracking of individuals in the field of security [1].

2 Methodology

2.1 User Registration

The user is required to provide its name and take pictures of their faces with the device's camera. The unique ID would be generated automatically and embeddings of the taken picture. The system stores user metadata locally on the device. This solution eliminates the requirement for centralized user databases while ensuring system security.

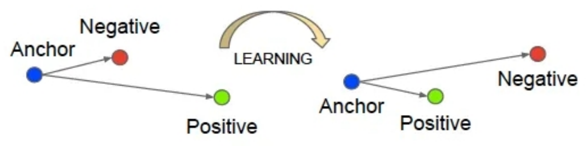
2.1.1 Pre-processing Face Verification

1. Face Alignment and Crop We utilized MTCNN(Multi-Task Cascaded Convolutional Networks) for facial detection to align and crop the user's face. It is a deep learning-based technique for face detection and alignment that locates and detects faces in digital photos or videos using a cascading sequence of convolutional neural networks (CNNs).

The MTCNN algorithm consists of three main steps: **The proposal network (P-Net):**The proposal network (P-Net) generates a set of possible bounding boxes that may contain a face. By applying several convolutional filters to the input image, the P-Net generates a collection of feature maps. These feature maps are then processed by a set of fully connected layers that calculate the probability that a face will show up in each region of the image. **Refinement Network (R-Net):** The candidate bounding boxes produced by the P-Net are refined by the R-Net, the second stage of the MTCNN algorithm. The R-Net crops the relevant areas of the input image using the candidate bounding boxes. To determine whether a bounding box is a face or not, these cropped areas are then resized to a predetermined size and run through a number of convolutional and fully connected layers. To pinpoint the precise location of the identified face, the R-Net additionally regresses the bounding box's coordinates.

Output Network (O-Net): The final stage of the MTCNN algorithm, the O-Net, further refines the bounding boxes and extracts the facial landmarks. The R-Net's optimized bounding boxes are used by the O-Net to crop the pertinent portions of the input image. These cropped regions are then resized to a preset size and passed through several convolutional and fully connected layers to ascertain whether or not a bounding box is a face. To further pinpoint the location of the identified face, the O-Net also extracts the coordinates of five facial landmarks, such as the mouth, nose, and two eyes, and regresses the bounding box coordinates.

2. Feature Extraction Using MobileFaceNet FR Model and Face Matching Facenet is a deep neural network that has been trained to map faces into a high-dimensional space, with the faces of different people far apart and those of the same person close together. Three images are used as input by the network's triplet loss function: an anchor image, a positive image (an additional image of the anchor), and a negative image (an image of a different person). In high-dimensional space, the triplet loss function seeks to maximize the distance between the anchor and negative images while minimizing the distance between the anchor and positive images.



Triplet Loss minimize the distance between an anchor and positive, while maximize the distance with negative (from Facenet)

Figure 1: Triplet Loss Function

Facenet makes use of the Inception ResNet architecture, which blends residual connections with the Inception module. The network can learn and produce face representations in the form of numerical embeddings thanks to this architecture. The distinctive characteristics of a person's face are represented by these high-dimensional vector embeddings. Face recognition software uses cosine similarity to identify whether or not two faces are those of the same person. In the code, we have set a threshold of 0.3 for cosine similarity to decide if the face matches an existing user; otherwise, the application reports "Face not matched."

3 Result

As per the shown below results on last page, the model successfully detects two different persons along with successfully registering the other person named "HCML" into the model. Unfortunately, the model counldnt be loaded into an Mobile Application , hence we used cloud based application to make the model working. Currently, the application works on EXPO-GO, along with the processing is being evaluated backend through FLASK-API.

4 Conclusion

We tried to created a privacy-preserving, real-time face verification model. This method improves security and user ease by removing the requirement for external servers or network access. Because of its ease of use, face recognition technology has become widely employed in security and finance and many other real-time applications. Face applications will become more sophisticated and diversified as science and technology advance. Future work will focus on gaining a deeper knowledge of the error cases in order to improve the model and application. Unfortunately, we were not able to complete the application as explained above.Currently, it works on Mobile Application connecting it through a server.

References

[1] Li, S., Deng, W., & Du, J. (2020). Deep Facial Expression Recognition: A Survey. *IEEE Transactions on Affective Computing*, 1-1. https://doi.org/10.1109/TAFFC.2020.2994251

[2] Wen, Y., Zhang, K., Li, Z., & Qiao, Y. (2016). A Discriminative Feature Learning Approach for Deep Face Recognition. In *European Conference on Computer Vision (ECCV).*

[3] Timesler, J. facenet-pytorch. GitHub repository, https://github.com/timesler/facenet-pytorch/blob/master/README.md.

[4] Dbtrs, F. ExFaceGAN. GitHub repository, https://github.com/fdbtrs/ExFaceGAN/blob/main/README.md.

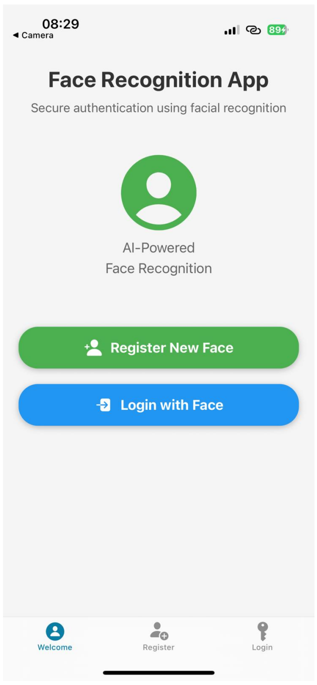
[5] Singha, S., & Prasad, S.V.A.V. (2020). Techniques and Challenges of Face Recognition: A Critical Review. *Journal/Conference.*

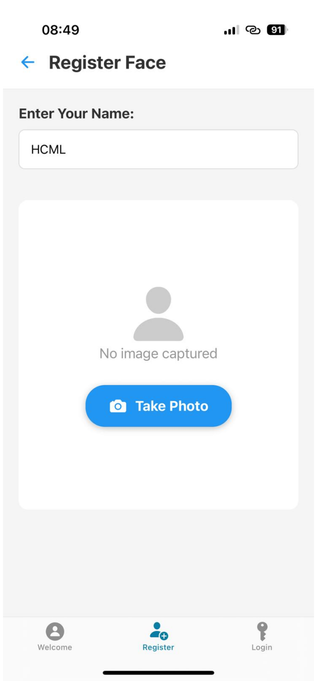
[6] Wang, M., & Deng, W. (2018). Deep Face Recognition: A Survey. *Neurocomputing*, 201, 15–28.

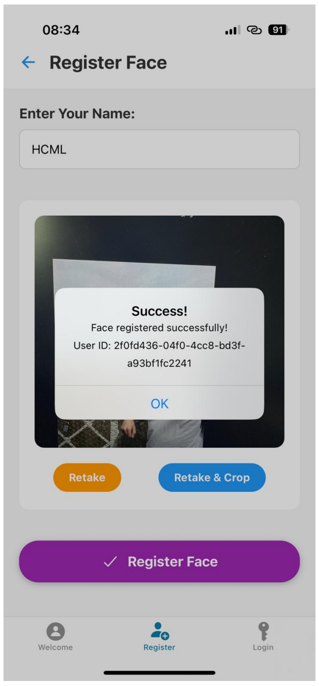
[7] Hassanpour, A. (2020). Lightweight Face Recognition: An Improved MobileFaceNet Model. arXiv preprint arXiv:2008.12358.

[8] Chen, S., Liu, Y., Gao, X., & Han, Z. (2018). MobileFaceNets: Efficient CNNs for Accurate Real-Time Face Verification on Mobile Devices. arXiv preprint arXiv:1804.07573.

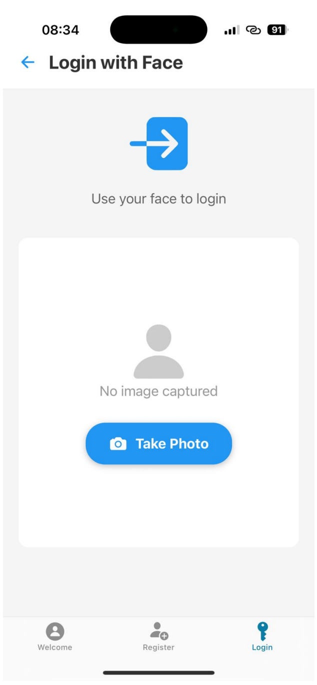
[9] Chan, K. Real-Time Facial Recognition with PyTorch FaceNet. Medium article, https://kean-chan.medium.com/real-time-facial-recognition-with-pytorch-facenet-ca3f6a510816.

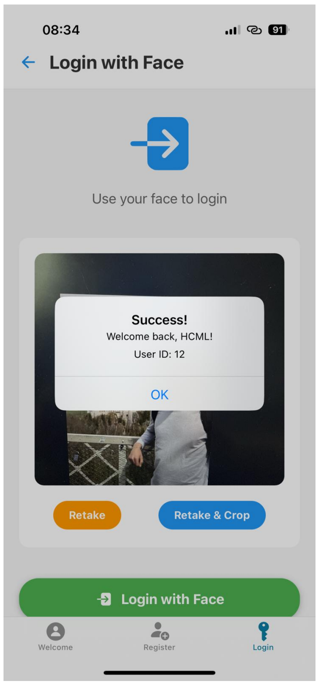






Success!
Face registered successfully!
User ID: 2f0fd436-04f0-4cc8-bd3f-a93bf1fc2241
OK





Success!
Welcome back, HCML!
User ID: 12
OK