Real-time Face Recognition using Deep Learning

Cheng Zhu, Dr. Sofya Poger

B.S. in Computer Science, School of Art and Science, Felician Univeristy

ABSTRACT

During the past few years, deep Convolutional Neural Networks (CNNs) have achieved promising results in face recognition. The state-of-the-art performance of face recognition using deep learning techniques combined with large datasets have surpassed human level performance on the Labeled in the Wild (LFW) benchmark [1][2]. Today, there is a huge demand for efficient real-time face recognition systems on mobile devices due to increasing mobile use in human life. For a mobile device, a face recognition system needs a tradeoff between accuracy and speed under constraints of its computational power. In this project, we deployed a real-time face recognition system on Raspberry Pi 4 using deep learning techniques from OpenFace [3]. The system implemented a conventional face detection method, histogram of oriented gradients (HOG) with linear support vector machines (SVM), to detect human faces. The detected faces were fed to a deep neural network pre-trained by OpenFace to compute 128-d face recognition embeddings as feature vectors for face classification. In this work, we explored the applicability of recent deep learning techniques to mobile devices with limited computational resources.

Keywords-Face recognition, face detection, deep learning, Raspberry Pi, CNN, OpenFace

INTRODUCTION

Face recognition refers to the technology that identifies or verifies human subjects in images or videos. It has been widely used in many areas, such as military, finance, public security and daily life as a remarkable biometric technique for identity authentication. In the past decades, there have been numerous approaches to implement face recognition with promising results, but many of their classifications require considerable computational resources. In recent years, the state-of-the-art performance in face recognition tasks has been achieved by deep convolutional neural networks which have been dominated by large scale datasets [1], [2]. Training of a deep neural network is often executed offline and in batch because it is computationally expensive to work with industry- or government-scale data size. However, the classification itself is cheap because a classifier produced by training can make predictions for each image in hundreds of milliseconds. Thus, in the scenarios that the target groups to classify do not often change, there is little concern about the time of training new classification models.

However, in mobile scenarios, a user may have a device performing real-time face recognition and may encounter different problems and requirements. For instance, the person of interest who the user wants to recognize may vary in different situations. If the user attends a meetup, the system should quickly learn to recognize the user's new friends. Many people in the mobile face recognition system are transient and the user only needs to recognize them for a short period of time. Therefore, the time to train new classification models in these scenarios becomes significant because there can be human subjects that are added or removed from the system.

In this project, we exploited the asymmetry of CNNs, where only training consumes computational resources, but classification runs cheaply. Our experiments showed that fast and accurate face recognition could also be achieved on small computers with limited computational capability such as Raspberry Pi.

METHODOLOGY

We used the deep neutral network from OpenFace library based on FaceNet [4] with some modifications to build an accurate and fast face recognition system on a tiny low-cost computer: Raspberry-Pi. In this section, we discuss the design and implementation of our proposed face recognition system. The workflow of our proposed face recognition system is shown in Figure 1.

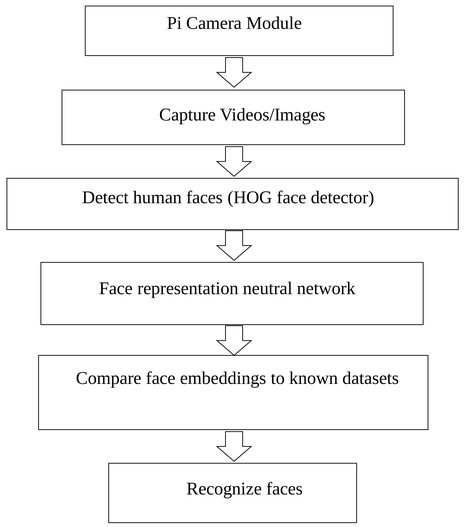

Figure 1. Workflow of our proposed face recognition system

We used a Pi Camera Module v2 connected to Raspberry Pi board to capture images and videos. A little portable display IPS screen is used to display the video and recognition results. The steps of running our purposed system can be categorized as:

1. A camera module of Raspberry Pi 4 captures a real-time video or pictures. The captured video/pictures will become input frames in face detection section.
2. In face detection section, for each image frame, we use the HOG algorithm to encode it and create a simplified version. The part of the image that mostly matches a generic HOG encoding of a face will be found upon the simplified version.
3. We figure out the pose of the face by finding the main landmarks in the face. The landmarks will be used to affine the image so that the eyes and mouth are centered.
4. We pass the centered face image through a neural network that knows how to compute 128 dimensional embeddings of each face as feature vectors.
5. At last, we compare all of the faces that we have measured to see which person has the closest feature measurements to the face we want to recognize.

In face detection phase, we applied HOG + Linear SVM face detector from dlib library [5]. The face detector will return a list of bounding boxes around the faces.

PREPROCESSING

Before face recognition phase, we need to do preprocessing work for face images. We used a face landmark detector from dlib library to detect the 68 landmarks for each face. Upon the large blue mean landmarks, our affine transformation makes the eye corners and nose close to the mean locations. Also, the affine transformation resizes and crops the image to the edges of the landmarks so that the input image to the neural network is 96 × 96 pixels. Figure 3 presents a real image in our dataset that is preprocessed.

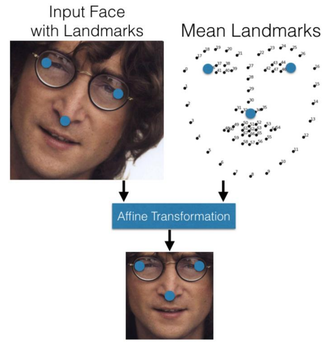

Figure 2. OpenFace's affine transformation [3]

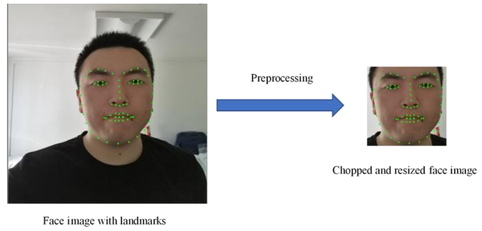

Figure 3. Preprocessing of a real image in our dataset

FACE REPRESENTATION NEUTRAL NETWORK

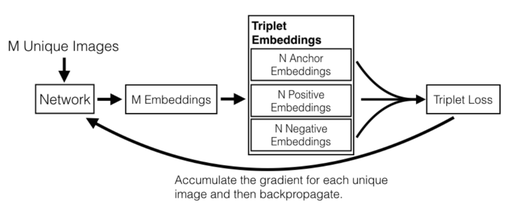

Figure 4. OpenFace's end-to-end network training flow [3]

Figure 4 shows how the networks of OpenFace are trained. Unique images from a single network are mapped into triplets. The gradient of the triplet loss is backpropagated back through the mapping to the unique images. The intuition behind triplet loss function is that we want our anchor image (image of person A) to be closer to positive images (all the images of person A) as compared to negative images (all the other images). In other words, we can say that we want the distances between the embedding of our anchor image and the embeddings of our positive images to be lesser as compared to the distances between embedding of our anchor image and embeddings of our negative images.

EVALUATION & SUMMARY

In this section, we present evaluations that measure the classification accuracies and training time of our proposed face recognition system on a subset of LFW images [6]. We compared our techniques with eigenfaces and LBPH from OpenCV library [7] on Raspberry Pi 4 4GB Model B with 1.5GHz 64-bit quad-core ARMv8 CPU (4GB RAM).

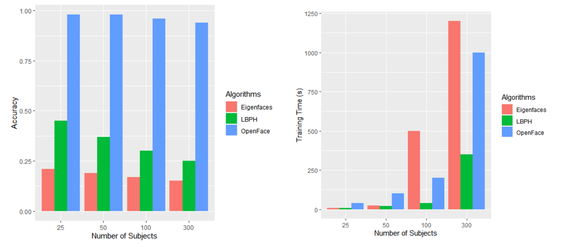

Figure 5. Evaluation of classification accuracy and training time

In summary, we fulfilled our objective to implement a real-time face recognition system on wearable devices in this project. We used face embeddings from FaceNet as feature vectors to train the region of interest (ROI). HOG algorithm helped us detect the faces in the specified input frames. The proposed techniques from OpenFace showed better results in classification accuracy and training time compared with other face recognition algorithms. The performance of our face recognition system on Raspberry Pi 4 can still be improved through future work:

1. Since our HOG + linear SVM face detector is not highly sensitive to profiles, in the detection phase, we can improve our detection performance by applying more advanced detection techniques for multi-pose faces.
2. The real-time recognition performance can be improved if we compute the face embeddings once every N frames (where N is user-defined variable) and then apply simple tracking algorithms to track the detected faces.
3. To further improve the abilities of the computing phase, external devices which can help Raspberry Pi increase computational power like Intel Neural Compute Stick can be used.

REFERENCE

1. Y. Taigman, M. Yang, M. Ranzato and L. Wolf, "DeepFace: Closing the Gap to Human-Level Performance in Face Verification," 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, 2014, pp. 1701-1708.
2. [2] S. Balaban, "Deep Learning and Face Recognition: The State of the Art," Biometric and Surveillance Technology for Human and Activity Identification XII (2015): n. pag. Crossref. Web.
3. Amos, Brandon et al. "OpenFace: A general-purpose face recognition library with mobile applications." (2016).
4. F. Schroff, D. Kalenichenko and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, 2015, pp. 815-823.
5. Davis E King. Dlib-ml: A machine learning toolkit. The Journal of Machine Learning Research, 10:1755–1758, 2009.
6. Gary B Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, Technical Report 07-49, University of Massachusetts, Amherst, 2007.
7. Gary Bradski et al. The opencv library. Doctor Dobbs Journal, 25(11):120–126, 2000.