Person Recognition for Access Logging - Phase II Report

Andrew Boka, REU Intern & Brendan Morris, Ph.D., Associate Professor
Department of Electrical & Computer Engineering, University of Nevada Las Vegas
Las Vegas, NV

Abstract/Overview

The goal of this project is to develop a complete hardware and software system for access monitoring into a secured facility using modern facial recognition technology (FRT) and lightweight, inexpensive components. FRT detects and recognizes individuals as they enter and exit the secured area, providing accessible logs for a visualization of this activity. The project uses Facenet FRT which takes a deep-learning approach to increase facial recognition accuracy. With input from a portable camera, the Facenet FRT is implemented on a Raspberry Pi using the Intel Movidius Neural compute system.

Introduction

Most security systems rely on access control within a secured area. Usually this access control will manifest itself in the form of gate guards or key card technology. Effective implementation of these systems comes with the overhead of complex record management as well as significant investment in personnel or key/badge technology. An effective key card system may also be compromised in the event that a card is lost or stolen. However, biometric identifiers tied to a particular individual's physiological characteristics do not have the same potential for misuse or exploitation [1].

Although historically undesirable for utilization in security or authentication systems due to high levels of inaccuracy, recent advances in facial recognition technology (FRT) have prompted its use in these types of applications. Some examples include a Windows 10 login option based in FRT as well as selfie-based authentication on smartphones and various banking applications [1]. In addition to their use in authentication, the potential for unique identification using biometric identifiers has been demonstrated with companies like Facebook and other social media platforms that are able to organize large amounts of image data using facial recognition.

The goal of the current project was to develop an independent, lightweight system for access monitoring and logging utilizing modern FRT. The envisioned system included a hardware/software system to monitor access at various entry and exit points as well as provide a full visual display of the activity. Rather than maintaining a database of users, the system dynamically populates a new registry of users daily. The purpose of this dynamic population of a database is to ensure that the system may be used for general purpose access monitoring to keep track of all individuals in a given facility. The software is currently implemented in hardware on a low-power processor which simulates actual "in-door" operation while being relatively compact and mobile for various testing applications. A web-based platform for visualization was

also developed and works alongside the hardware system to provide real-time updates on access information as well as the system status. The current project expands upon an earlier lightweight prototype through its implementation of a more modern facial recognition system.

Previous Work/Background

Phase 1 Prototype

Prior to the current project, a lightweight access monitoring system was developed by Morris [1]. The lightweight system operated at 30 fps on a laptop or 10 fps on a Raspberry Pi using Eigenfaces for recognition with a web-based visualization as an overview of daily access activity (Figure 1). However, recognition accuracy was unacceptable requiring the use of more modern deep-learning recognition techniques. The open source project, OpenFace, was explored for improved recognition accuracy with a deep learning approach. Unfortunately, OpenFace required significant resources limiting its portability.

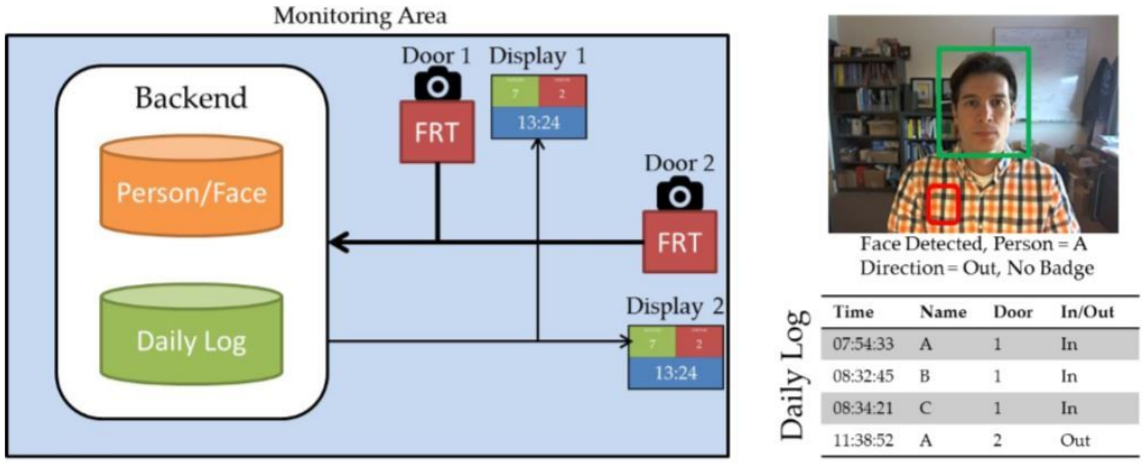

Figure 1. Original Facial Recognition Prototype developed by Morris [1].

Eigenfaces

Morris's lightweight, low-power prototype uses a facial recognition technique known as Eigenfaces [2]. The technique utilizes principal component analysis (PCA) to project face images onto a feature space which spans the significant variations among known faces.

Each face image of size 128 x 128 pixels can be considered as a point in 16384-dimensional space. Principal Component analysis is used to determine the vectors which best account for the distribution of face images within the image space and these vectors are then used to define a subspace of face images known as "face space." These vectors are referred to as "eigenfaces" because these vectors are eigenvectors of the covariance matrix corresponding to the original face images and due their appearance being similar to that of a face. The face space is learned using the Caltech Faces 1999 database which is composed of 450 images of 27 different people. Recognition is reduced to more of a verification problem since a k-nearest neighbor search of a detected face with all stored faces in the system is used for this process.

The biggest drawback of the Eigenfaces technique is the underlying assumption that the faces appear in the same frontal pose and under the same lighting conditions. This is inadequate for use in a facial recognition system where lighting and pose are not generally consistent factors. Although Morris's full-scale system using the OpenFace implementation produced more accurate results than the Eigenfaces software, it could not be maintained on the lightweight, low-powered hardware.

Facenet

The Facenet [3] system utilizes a more modern, deep learning approach to the issue of face recognition. Given an input image, the convolutional network is then used to learn an embedding for the image which has properties conducive to facial recognition, verification and clustering. As a result of the loss function used, these embeddings have the unique property that the Euclidean distance between them is directly proportional to face similarity. As a result, face verification problem is reduced to a task of thresholding a distance between two embeddings and recognition becomes a kNN classification problem. This method is far more direct, efficient and practical than previous facial recognition approaches based on deep networks.

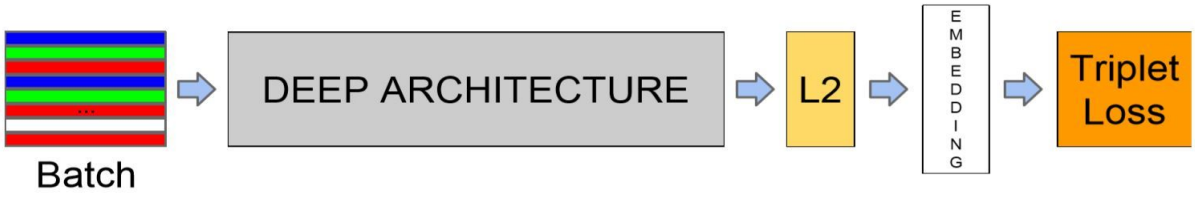

Figure 2.Scroff et al."s [3] Facenet Model structure. The network consists of a batch input layer and a deep CNN followed by L2 normalization, which results in the face embedding. This is followed by the triplet loss during training .

The deep network structure consists of a batch input and a deep CNN which is followed by unit normalization resulting in the facial embedding. The normalization in this procedure ensures that the otherwise unconstrained facial embeddings are constrained to live on the unit hypersphere which allows for a generic embedding space to be constructed and used on previously unseen faces. Triplet loss training directly precedes the L2 normalization layer during training and is responsible for the minimization of distances between the same faces and the maximization of distances between different faces. Triplet loss presents faces to the deep learning algorithm in a set with an anchor image, a positive image and a negative image. The anchor and positive

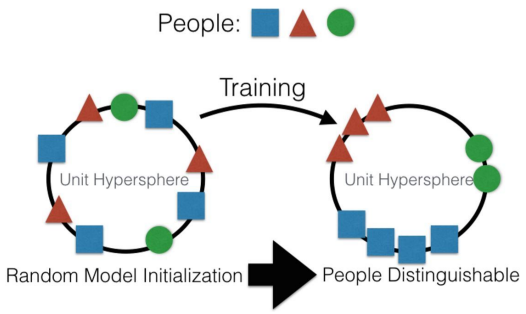

Figure 3. Illustration of FaceNet's triplet-loss training procedure [4].

face image are both from the same individual while the negative face image is from a different person. This type of loss ensures that the distance between the anchor and positive image is less than the distance between the anchor and negative image.

While previous deep learning approaches had been developed for facial recognition, the Facenet implementation gives higher classification accuracy on benchmark datasets as well as reducing error rates by up to 30 percent [3].

Current Project Design/Implementation

The current project was able to overcome shortcomings of the Eigenface FRT by redesigning the lightweight, low-power access monitoring prototype using Facenet technology.

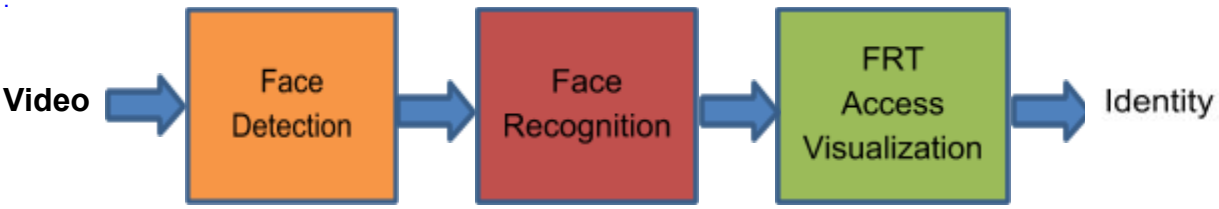

Figure 4. The current project's facial recognition model structure

Face Detection

The full method of facial detection used in the implementation of the facial recognition system on the Raspberry Pi is very similar to the OpenFace process [4]. There are 4 main steps in this process: 1. Face detection; 2. Alignment of detected faces; 3.Use of deep neural network to embed the face onto a 128-dimensional unit hypersphere; and 4. Perform verification, recognition or clustering tasks using the generated embeddings.

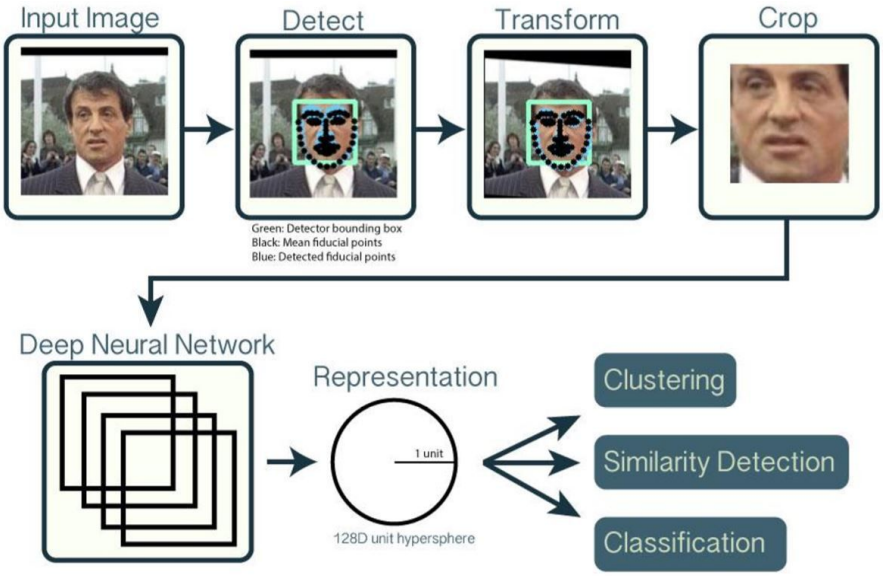

Figure 5. OpenFace deep neural network implementation for face recognition [5].

The system uses a Haar-feature based approach to face detection which has proved to be very efficient and run well on a Raspberry Pi [6]. A Haar feature considers pixel intensities in certain regions and is considered a weak classifier for identification. Adaboost training is used to select the relevant Haar features and these are arranged into a strong cascade classifier for object detection. More advanced detection algorithms were considered and subsequently rejected due to high computational demand.

Face Recognition

Prior to input into the network, a detected face image must undergo a process of pre-alignment. One of the biggest challenges in facial recognition is learning to create a model which is insensitive to variations in lighting and pose between images. While previous methods were prone to error under conditions of greater variance [2], the pre-alignment process provides a normalization factor for these conditions. Once a face has been detected using a Haar-feature based approach, 68 facial landmarks are mapped to the face and alignment is performed. The alignment process uses affine image transformations to align various facial landmarks such that they appear in consistent relative positions across images.

Once the image has been aligned, it is ready to be passed through our deep neural network to generate a low-dimensional embedding representation of the face. The neural network used is a TensorFlow implementation of the network described in the Schroff et al. paper [3]. The embedding is subsequently compared against previously seen embeddings using a k-NN classification system in addition to a threshold distance which allows us to classify seen and unseen faces before entry into a database.

The prototype system runs in real time and provides output from real time video processing which displays bounding boxes around the detected faces and names next to these bounding boxes as seen in figure X. In a real application, this type of visual output would be unnecessary since the system would be embedded in a door or other structure and the small touch screen display would be used for the sole purpose of providing relevant access information. The system will also need to independently identify when a user has entered a room rather than being prompted by user input. This will most likely require the implementation of additional detection systems to allow the system to determine when an entry or exit is occurring.

A more realistic version of the prototype system was also developed which simply displays bounding boxes around detected faces but does not attempt to label these faces. It is not necessary for identification to be continuously performed every frame on every face. It is only necessary to perform identification in the event that an entrance or exit has been performed. As the current k-NN classification is slow due to the high dimensionality of the data, it is also highly impractical to perform identifications every frame. In order to increase accuracy of the face recognition, there is no persistence or attempt to use correlative or other tracking techniques on individual faces. Instead, faces are simply detected each frame independent of previous frames.

FRT Access Visualization Implementation

The final component of the access monitoring system is data management and visualization, of which there are various types provided. The first visualization is intended as a quick access look at the system status - number of people that have entered, exited and remain in the monitored area - and it intended to be shown on an “in-door” display which is mounted directly onto the camera hardware.

Figure 6. Management log interfaces are shown. The Event Log on the Left provides a time ordered summary of everything that has occurred during the day The Current Occupant Log on the Right gives a summary of who is currently in (green) and who has left (red) the system [1].

A more complete visualization intended as a system management interface is built on html web technology in order to provide cross platform accessibility. Through the use of WebSocket technology, the “in-door” system is able to communicate in a bidirectional manner with the html web page. WebSockets [7] performance is a result of the protocol’s duplex nature as well as the fact that it does not rely on HTTP communications. Muller describes the WebSocket protocol as “[enabling] two-way communication between a client running untrusted code in a controlled environment to a remote host that has opted-in to communications from that code…” The protocol consists of an opening handshake followed by basic message framing, layered over TCP. The goal of this technology is to provide a mechanism for browser-based applications that

need two-way communication with servers that does not rely on opening multiple HTTP connections." The initial HTTP connection is replaced by a WebSocket connection which uses the same underlying Transmission Control Protocol (TCP) which ensures the consistency of the data while reducing the overhead traditionally associated with HTTP requests.

The "more complete" visualization mentioned above provides an Event Log and Current Occupant Log which can provide a more complete overview of all the events which have occurred throughout the access monitoring system. The Event Log provides all daily information including a face image, names, time, location and type of action (entrance or exit). The Current Occupant Log provides a face image, name and time of action (entrance or exit) for a given location.

Movidius Stick Implementation

The Intel Movidius Neural Compute Stick (NCS) is a low-power device which allows the implementation and optimization of large deep-learning models in lightweight applications. It is important to note that the Neural Compute Stick does not support training these models and this task must be done separately before deployment onto the NCS. The power of the NCS comes from the Myriad 2 Vision Processing Unit (VPU), which is essentially a fanless GPU running on a USB. Currently, the device supports Tensorflow and Caffe based deep neural networks. For this project a Tensorflow implementation of Facenet was deployed on the NCS. More specifically, a pre-trained facenet network is first converted to a format suitable for compilation on the NCS and then this converted network is compiled into a .graph file to be used with the NCS device. The NCS SDK comes with a Python API which provides all the necessary tools for utilizing the power of the NCS in the the FRT system.

Experimental Evaluation

The current prototype system utilizes two cameras to simulate a single point of entry and a keyboard trigger to simulate an entry or exit. Upon user request for simulation of an entry/exit the users detected are given generic names with the following convention: XXXX_YYY. The XXXX is used to indicate the identity of the person. YYY if the occurrence count or number of times the particular user has been seen since deployment of the system or a daily reset of the database. As an example, 0001_010 would indicate the tenth occurence of person 1 [1].

The earlier FRT system faced major performance issues due to the use of EigenFace recognition. This is a technique which adapts poorly to variations in lighting or pose between two face images. Additionally, its focus on face structure often causes confusion between individuals who have similar facial features such as a nose or beard. For the access monitoring system, it is important to adapt to changes in lighting conditions or pose since two separate cameras are used for entrance and exit monitoring. The Facenet implementation has provided an adequate replacement for the antiquated EigenFace model. The newer model is able to achieve around 99.6% accuracy on the popular Labeled Faces in the Wild (LFW) [3] dataset while significantly reducing the error rate. The commercial production of the Movidius Neural

Compute Stick (NCS) has made it possible to deploy deep neural networks on lightweight, low-power applications through the use of its Vision Processing Unit (VPU). This allows the full system to run on a Raspberry Pi while achieving optimal performance.

Conclusion

This report details the implementation of a complete access monitoring system utilizing facial recognition technology. The specific contribution of this project was to replace the obsolete Eigenface recognition used on an earlier lightweight, low-power prototype with a much more effective deep learning facial recognition algorithm. The prototype includes various low-power devices including a Raspberry Pi, two cameras and a Movidius Neural Compute Stick (NCS). Using the Vision Processing Unit (VPU) in the NCS, it is possible to efficiently run an implementation of the Facenet [3] algorithm on the lightweight prototype. In contrast to the Eigenface method which responded poorly to changes in lighting or pose, the deep learning approach to facial recognition is not sensitive to this sort of variability. In order to further develop the system, it will be necessary to remove intervention for face capture and introduce support for multiple units in order to effectively manage larger, more complex areas [1].

References

[1] Brendan Morris, “Person Recognition for Access Logging - Phase I Report,” UNLV Department of Electrical and Computer Engineering, 2016.

[2] Matthew Turk and Alex Pentland, "Eigenfaces for Recognition," Journal of Cognitive Neuroscience, vol. 3, no. 1, pp. 71-86, 1991.

[3] Florian Schroff, Dmitry Kalenichenko, and James Philbin, "FaceNet: A Unified Embedding for Face Recognition and Clustering," in Computer Vision and Pattern Recognition, 2015, pp. 815-823.

[4] Brandon Amos, Bartosz Ludwiczuk, and Mahadev Satyanarayanan, "OpenFace: A General Purpose Face Recognition," CMU School of Computer Science, CMU-CS-16-118, 2016.

[5] Brandon Amos, “OpenFace: Free and Open Source Face Recognition with Deep Neural Networks,” Carnegie Mellon University,2016. http://cmusatyalab.github.io/openface/

[6] Paul Viola and Michael J Jones, "Robust Real-Time Face Detection," Int. J. Comput. Vision, vol. 57, no. 2, pp. 137-154, May 2004.

[7] Gabriel L. Muller, “HTML5 WebSocket protocol and its application to distributed computing,” Cranfield University School of Engineering, 2014. https://arxiv.org/pdf/1409.3367.pdf