Deep Learning Based Face Recognition System with Smart Glasses

Ovidiu Daescu
Department of Computer Science
The University of Texas at Dallas
Richardson, Texas
daescu@utdallas.edu

Hongyao Huang
Department of Computer Science
The University of Texas at Dallas
Richardson, Texas
hhuang@utdallas.edu

Maxwell Weinzierl
Department of Computer Science
The University of Texas at Dallas
Richardson, Texas
maw150130@utdallas.edu

ABSTRACT

Individuals with prosopagnosia have difficulty in identifying different people by their faces. Our goal is to design and develop a face recognition system with wearable glasses to recognize faces and provide identity information to users. Unlike other existing systems that run locally on glasses or cellphones, we introduce a client-server architecture system for facial identification. We designed and implemented applications both on a pair of smart glasses and a cellphone to capture images and communicate with the server. Deep Convolutional Neural Networks (CNN) were chosen to build our face recognition on the back-end system and we achieved 98.18% accuracy for face recognition. The system is designed to handle new identities and new faces without having to rebuild the model.

CCS CONCEPTS

• **Human-centered computing** → **Accessibility**; • **Computing methodologies** → **Neural networks**.

KEYWORDS

Convolutional Neural Network, wearable device, face recognition, augmented reality, prosopagnosia

ACM Reference Format:
Ovidiu Daescu, Hongyao Huang, and Maxwell Weinzierl. 2019. Deep Learning Based Face Recognition System with Smart Glasses. In *The 12th PErvasive Technologies Related to Assistive Environments Conference (PETRA'19), June 5–7, 2019, Rhodes, Greece.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3316782.3316795

1 INTRODUCTION

When participating in a social event, recognizing the identity of a person is the first step to engaging in a face-to-face conversation. Likewise, when meeting unknown or unfamiliar people, we often want to know their identity along with other information such as relation, age, or job. In social situations, it can be embarrassing forgetting the name of someone you know. Moreover, prosopagnosia, or "face blindness", a cognitive disorder resulting in the inability to recognize familiar faces [12], is suggested to affect up to 2.5% of the U.S. population [3][8]. Hence, it has become an interesting research topic to develop a wearable system that can aid visual memory for individuals, especially for helping to improve the social life for people who are suffering from prosopagnosia.

With the emergence of smart glasses, we can enhance the information available in real-world social environments. Smart glasses often consist of a wearable display, a camera which provides a similar viewing angle to the viewing angle of your eyes, and connecting capability to other devices. It then becomes possible to build a system to perform real-time face recognition tasks with the help of smart glasses. Our system combines modern computer vision and machine learning models with wearable technologies. The system is general enough such that it does not require re-training for new users or new identities while also being fast enough to run in real-time.

1.1 Our Contributions

The computational efficiency is important for applications on smart glasses and mobile devices. We introduce a server based model to carry out the image processing and face recognition tasks. With the arrival of 5G technologies, we will be able to achieve better computational efficiency on data transmission across wearable glasses, mobile devices, and servers. Our glasses-phone-server framework runs the following process. We use the cellphone as a controller to trigger the camera on smart glasses to capture facial images of the person of interest. The captured images are sent to the cellphone and then uploaded to the server through the cellphone application. The face detection and recognition tasks are carried out by our model running on a server. The identity of the person is retrieved from server and displayed on the cellphone and glasses. On the back-end side, we trained a deep convolutional neural network (CNN) to perform the face recognition task. We also built a real-time system to use the model in a video stream and set it up to run on both a Raspberry Pi and a laptop. This local application displays a bounding box on the face along with a label with the identity. It also allows the user to add new identities and new images for existing identities.

We adapted and modified the architecture of the face recognition system in [13] to deploy on our back-end system. Our first modification is that we used a more recent CNN, Inception-ResNet [14] in our back-end system. We also modified the neural network by utilizing the final layer as a L_2-normalized vector embedding layer instead of a final softmax layer in the original version of this network. The model we constructed on the back-end side is able to create a mapping between the facial image of a person and the

PETRA'19, June 5–7, 2019, Rhodes, Greece

ACM ISBN 978-1-4503-6232-0/19/06…$15.00
https://doi.org/10.1145/3316782.3316795

embedding of their identity. After the training phase, the neural network is able to create this mapping and generate an embedding for every facial image. A centroid vector is also created for each identity based on all embeddings of the same identity and will be further used for recognition. For any unknown face, we allow users to assign an identity to it and upload this image along with the user-specified identity. The trained CNN first maps a new facial image provided by user to the embedding of the identity. Then the model recalculates the centroid vector of this identity. If it is a new identity, a new centroid vector is created. Therefore, the back-end system is able to learn these new faces without going through the whole training process.

For privacy concerns, we do not store actual images taken from the glasses on our back-end system. Instead, we only store facial embeddings extracted from images; it would be very difficult to reconstruct facial images from embeddings without direct access to the model. We discuss this further in Section 3.1. Additionally, for each user, an independent facial identity profile is stored so that any new identities added by a user will not be shared across all users. User requests for facial identification are only compared against identities the user has previously identified and saved.

We organize this paper as follows. The next section discusses related work. In Section 3, we present the system architecture and the implementation details of our face recognition system. Methodology and configurations to build our back-end face recognition model are described in Section 4. We also present the process of training the deep neural network in this section. Section 5 presents the performance of the face recognition model along with the evaluation of performance on the whole system. Finally, conclusions and future work are discussed in Section 6.

2 RELATED WORK

Face recognition is one of the most studied research problem in computer vision. Problems associated with face recognition have been studied by researchers from various areas for over four decades [19]. Recently, great improvements of face recognition have been presented with the help of deep learning, especially deep convolutional neural networks. Deep neural networks are not limited to facial identification problems, recent methods show improvement in constructing general models for the task of face verification and recognition. Schroff et. al [13] proposed FaceNet, a deep convolutional network based system, that can directly learn a mapping from face images to a measure of face similarity. A deep residual network was proposed by He et. al [4] for more general image recognition tasks. They presented a residual learning framework to reduce the training time of networks that are substantially deeper than those used previously. Szegedy et. al [14] introduced the residual connections in conjunction with a more traditional Inception architecture. The introduction of residual connections not only accelerated the training process of neural networks, but also improved the performance on many image recognition tasks.

In the meantime, researchers have been working on solutions to improve the social life for people affected by prosopagnosia. In the technology community, a few systems using variations of wearable devices have been reported in the past few years. Krishna et. al [9] presented a wearable facial recognition system to provide aids in social interactions for visually impaired individuals. Their system runs processing tasks on a nearby USB-connected computer which is not practical in real-world situations. The evaluation of face recognition task is limited to only 10 subjects' faces. In order to mobilize the recognition system, new systems involving smart phones and more powerful wearable devices have been presented. Wang et. al [18] pushed the image processing and face recognition task to cellphones. Images and results are transmitted to glasses via a video driver board. However, the system requires collecting images and training the algorithm prior to usage. Hence it is only able to identify faces included in the offline trained model. Mandal et. al [11] implemented the face recognition system on Google Glasses with the emphasis on identifying faces under different lighting and various angles of faces. Although their system has been evaluated on 88 subjects' faces, retraining of the model is still required for new identities which are not in the initial data set.

3 OVERVIEW OF FACE RECOGNITION SYSTEM

Our goal is to construct a system that can capture images through the smart glasses based on users' commands and then return the face recognition result back to user. The system needs to be general enough such that it does not require re-training for new users or new identities while also being fast enough to run in real-time. Hence, we propose a client-server architecture including a pair of smart glasses, a cellphone, and a back-end server to perform the face recognition task. Figure 1[1] gives the general structure and process of the whole system. The client side includes a pair of smart glasses and a cellphone connecting to the glasses through a Bluetooth connection. The server side system handles the image processing and the face recognition tasks.

On the client side, we designed applications both on the smart glasses and cellphone. The main face recognition process of client-side application goes as follows. When a user opens the application on both devices, the phone application will check the Bluetooth connection and ask the user to make connection with the glasses. The user then sends a request to the glasses asking to take a picture and the phone application retrieves the image captured on glasses. The phone application initiates an HTTP post request in the background and sends the captured image to the server. In the meantime, the user may become closer or have a better angle of the person of interest. Hence, we designed the application so that the user is able to make a new image-capturing request while the server is processing the previous image. The face recognition result is displayed on the screen of both the cellphone and the glasses as soon as the results are received by the client.

The computing power on wearable devices and cellphones is limited compared to a server. Running face recognition tasks on wearable devices also drastically reduces battery life. Besides, many face recognition systems are limited to only recognizing identities that were in the offline training set of the model. When dealing with unseen faces, they have to go through the training phase again to rebuild their face recognition model. In order to make our model more general, while fast enough to handle new faces, we chose to move the image processing and the face recognition task to a server

[1] https://draw.io/

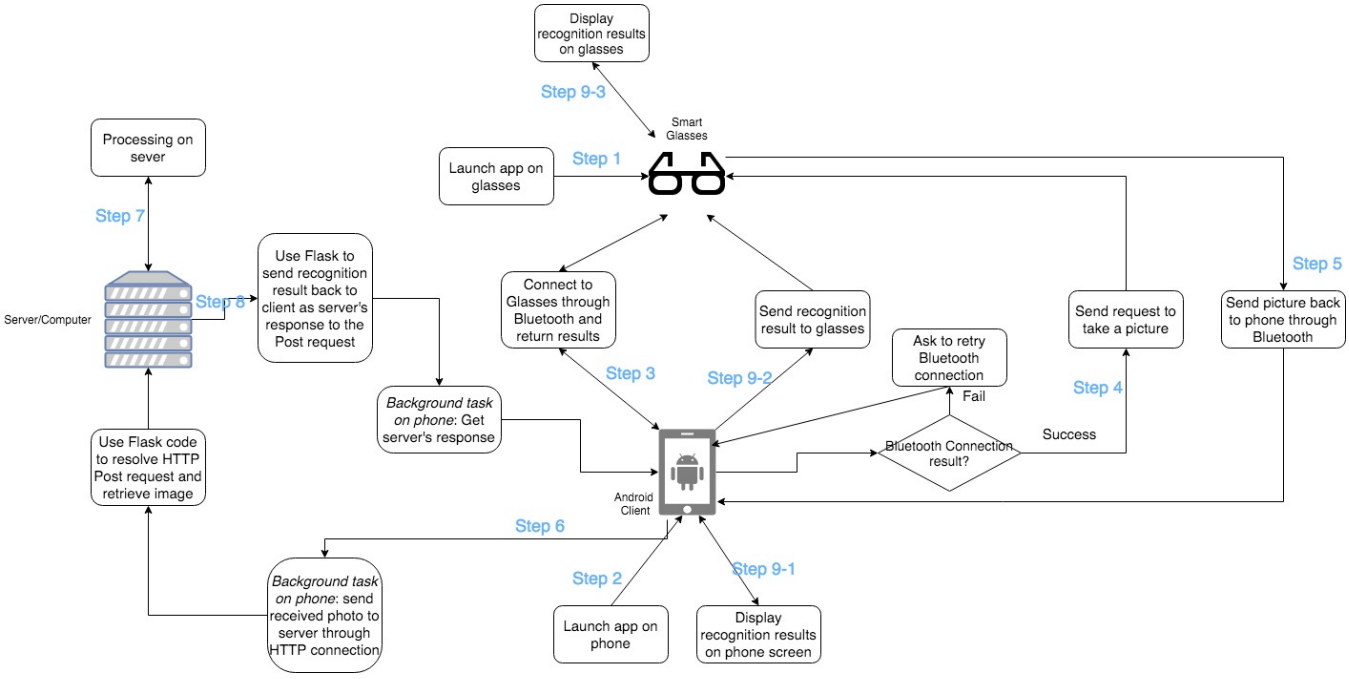


Figure 1: Structure of whole system design

and construct the model using a deep convolutional neural network. All image processing tasks are handled by our back-end system. Upon receiving a new image from the cellphone application, the system first runs face detection and crops facial regions detected in that image. All faces appearing in the image will be fed into the trained neural network for face recognition. Identity results of cropped facial regions are sent back as a server response to client's request.

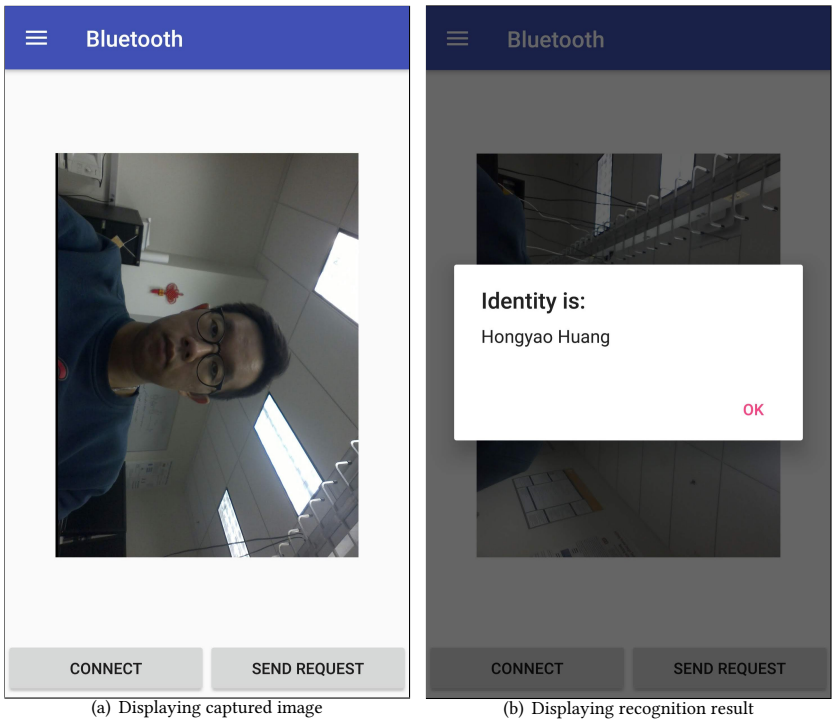
We adapted the architecture of our face recognition system from FaceNet [13] but rebuilt it with a different deep convolutional neural network based on Inception-ResNet [14]. The neural network produces a function which maps a cropped image of a person's face to a vector which represents the identity of that person. Additionally, an average (centroid) vector can be produced for each identity based on multiple images from different poses and backgrounds and this centroid vector maintains the same distance properties. Therefore, maintaining a independent centroid vector for each identity allows for an efficient method of determining the identity of a person. We can also determine a rejection distance by which a vector can be said to not belong to any of the known identities if it is further than that rejection distance from any of the known centroid vectors.

The neural network will first generate an embedding of the face image, which is a vector for the facial region inside that image. Then it will determine the identity based on the embedding's distance to every known identities' centroid vector for that particular user. As we mentioned in Section 1, users can use our application to add new identities along with multiple images. In this case, the neural network creates a new centroid vector of this user-specified identity. Each time a user uploads a new facial image with a known identity, the model first generates the embedding of this image. The model then calculates a new centroid vector of this identity by using this new embedding vector and other embedding vectors of the same identity. Hence, the system can gradually learn to recognize a face better and better as new images of the same person are added, improving the quality of the centroid vector for that identity.

3.1 Implementation of Client-Server Architecture

On the client side, we built an application on smart glasses and an application on an Android smart phone. We implemented two major functions to perform the face recognition task. The user will first be asked to connect the phone with smart glasses through Bluetooth connection. Once the connection succeeds, the user can start the "send request" session on phone which will send an image capturing request to connected glasses. Upon receiving the image from glasses, the image will immediately be sent to the server. Meanwhile, the user can make a new image capturing request on the foreground by pressing the "send request" button again. We set up a background thread using Android's "AsysncTask" API[2] to handle all communication between cellphone application and the server. The background thread is responsible for uploading the image to the server and receiving the identity result from server. We upload images to server with a HTTP Post request. A unique user ID will be assigned to each user and this user ID is embedded as a query string in the URL for the HTTP Post request. This ID lets the server know which user is making a request. The identity of the person in that image is then sent back to phone as the server's

[2]https://developer.android.com/reference/android/os/AsyncTask



(a) Displaying captured image (b) Displaying recognition result

Figure 2: Cellphone Application Interface

response to previous request. Once the result is received, we decode the byte string and display the result as a pop-up message dialog on both phone's and glasses' screens showing the person's identity to user. Figure 2a displays the image captured from glasses while figure 2b presents the pop-up dialog for displaying the result on both the phone and the glasses.

The smart glasses we used in this project are Epson's MOVERIO BT-300[3], an Android based smart glasses with Bluetooth. We utilized the camera on this wearable device to capture images. Capturing images on demand reduces the load of Bluetooth transmission between the cellphone and the glasses, which also has the benefit of reducing battery consumption when running the application. Hence, the camera on glasses only takes a picture when a user request is received. Images captured on the glasses have the resolution of 2560×1920 pixels. This resolution allows our face detector to detect faces up to around 15 feet from in-lab experiments. In order to reduce the load of Bluetooth communication, we compress images on the glasses to 85% JPEG quality before being sent to the phone.

We deployed our back-end system on both a Raspberry Pi and a laptop with Nvidia GTX1080 GPU. We utilized Flask[4], a Python based open source web framework, to handle the server-side HTTP requests in our implementation. We created an independent identity model for each user. Each identity model is not shared across the system. Only embedding vectors of facial images are stored on the server instead of actual images to maintain the identity model for each user. When a request from a client is received, the server system resolves the query URL to get user id so that recognition task is performed on this user's unique identity profile. Only the discovered identities are returned in the request response. Recent work by Mai et al. [10] shows that it is possible to train models to reconstruct face images from face embeddings, but this requires a method to produce embeddings to use for reverse-engineering images. This further justifies our system architecture, in that all the face embeddings and face embedding models can be secured on a centralized server, where only identity results are returned in the request response and no face embeddings or models are made public.

[3]https://tech.moverio.epson.com/en/bt-300/
[4]http://flask.pocoo.org/

4 FACE RECOGNITION MODEL

In this section, we discuss the model we applied and deployed to our back-end system for image processing and face recognition tasks. Figure 3 shows the structure of our back-end face recognition model. Figure 4 visualizes the process we used to build our model with a deep convolutional neural network. We present the convolutional neural network as a black box to learn and generate the face vector from the facial images fed into it. The motivation is to map cropped images of faces of dimension $length \times width \times colors$ to a d-dimensional vector which summarizes the features of that face. The model which produces this mapping can be trained to produce vectors which are close in Euclidean distance for identical identities while far in Euclidean distance for different identities. This allows a general model to be constructed and trained which produces this mapping on any given images of faces and identities, regardless of

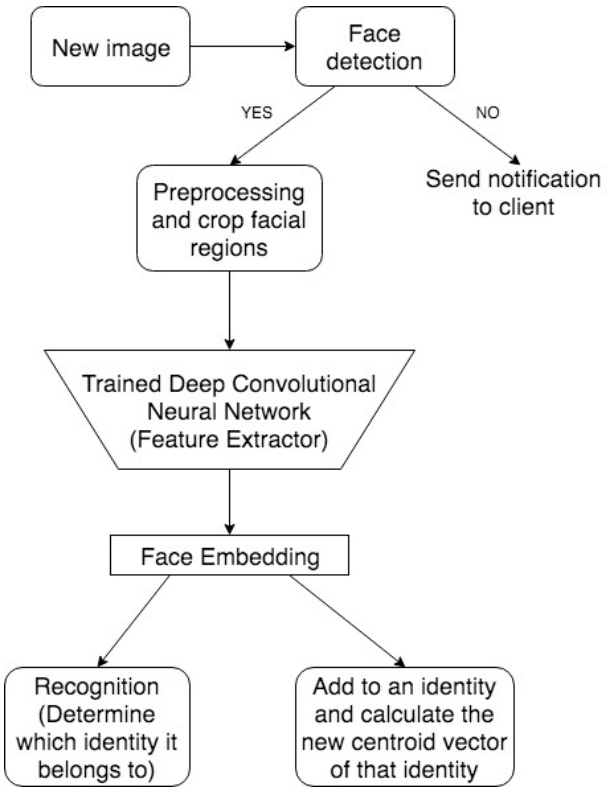


Figure 3: Structure of back-end face recognition system

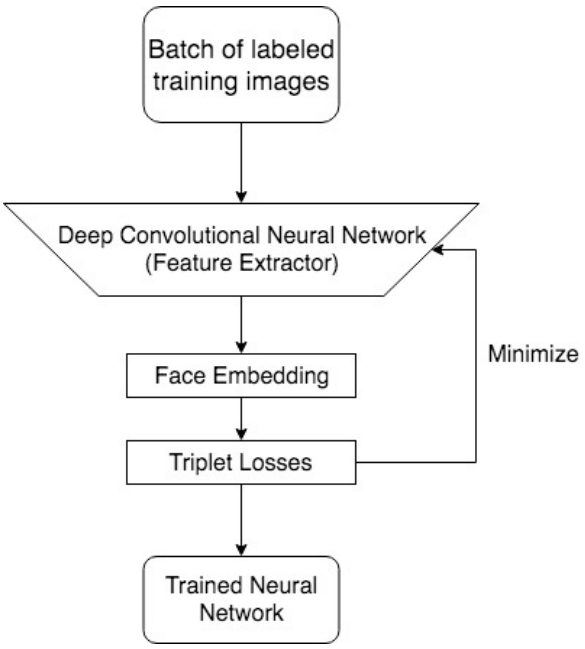


Figure 4: Training process structure

whether the model ever trained on these faces or identities. Once the face embedding is generated, the face recognition task becomes a classification problem.

The face vector, namely, an embedding $f(x)$, is a mapping from an image x into a feature space R^d, where $d = 128$, such that the squared Euclidean distance for all faces of the same identity, independent of imaging conditions, is small, while the distance between a pair of images from different identities is large. For the black box neural network to learn the ability to produce this mapping, we adapted triplet loss from FaceNet [13]. A triplet is defined by three elements produced by the neural network. In our case, these three elements are three face embedding vectors mapped from three facial region images of cropped face. As defined in [13], we want to ensure that an image of a specific person x_i^a *(anchor)* is closer to all other images of the same person x_i^p *(positive)* than images of any other person x_i^n *(negative)* by some margin α. We use $f(x_i^a)$ to represent the embedding of an anchor image. Similarly, $f(x_i^p)$ represents the embedding of a positive image and $f(x_i^n)$ represents the embedding of a negative image. The goal is to minimize the distance between $f(x_i^a)$ and $f(x_i^p)$, and maximize the distance between $f(x_i^a)$ and $f(x_i^n)$, for all possible triplets $(f(x_i^a), f(x_i^p), f(x_i^n))$ which violate the inequality in Equation 1 in the training set T.

Therefore, as defined in [13] Equation (2), we want,

$$\left\|f(x_i^a) - f(x_i^p)\right\|_2^2 + \alpha < \left\|f(x_i^a) - f(x_i^n)\right\|_2^2, \quad \forall (f(x_i^a), f(x_i^p), f(x_i^n)) \in T \tag{1}$$

where α is a margin that is enforced between positive and negative pairs.

The loss that is being minimized while satisfying the constraint in Equation 1 is,

$$L = \sum_i^N \left[||f(x_i^a) - f(x_i^p)||_2^2 - ||f(x_i^a) - f(x_i^n)||_2^2 + \alpha \right]_+ \tag{2}$$

After a face embedding of each image is generated, we compare the squared L_2-distance between different embeddings. In order to speed up the face recognition process, a centroid vector is produced for each identity. The centroid identity vector construction is similar to centroid vector construction for document summarization. This centroid vector can be constructed from multiple images of the same face with different poses and backgrounds. When recognizing the identity of a face image, our back-end system does not need to compare with every existing embedding. Instead, we only need to compare it with the centroid vectors and we can determine whether the face image belongs to an identity based on some selected distance threshold. This allows for an efficient method of determining the identity from a given image.

We constructed the deep convolutional neural network (CNN) based on a more recent Inception-ResNet [14] architecture. We use this architecture as a base because it introduces residual connections in conjunction with a more traditional inception architecture. It has been shown that the combination of these two methods can reduce the training time significantly over a GoogLeNet [15] based CNN, which is used in FaceNet [13]. This more recent network also

outperforms previous networks on image recognition tasks. Compared with the original Inception-ResNet architecture, we modified the input image size from $299 \times 299 \times 3$ to $96 \times 96 \times 3$. The combination of utilizing the residual connections in the Inception-ResNet architecture along with the reduction in input size allowed us to both fit the model in GPU memory while also speeding up the training process significantly. Our model trained to convergence in 240 hours on a single Nvidia GTX 1080 GPU, while the FaceNet [13] model took 1000 to 2000 hours to train on a CPU cluster. This modification allows the model to train much faster with only a minor reduction in accuracy (99.63% on LFW in FaceNet [13] vs our 98.18% on LFW). Our modification of the network does not have a final softmax layer for identity classification, like some previous deep learning based face recognition models such as DeepFace [16], but instead follows the same methodology as FaceNet [13] and utilizes that final layer as a L_2-normalized vector embedding layer which produces face embeddings.

4.1 Dataset and Preprocessing

Training the neural network requires significant data. We decided to utilize the VGGFace2 [1] dataset for training and the Labeled Faces in the wild (LFW) [6] dataset for our hold-out set. The VGGFace2 dataset contains a significant number of images of faces of different identities which makes it perfect for training a system for discerning between identities. The dataset also has significant variation in lighting and background environments as can be seen in Figure 5 which helps the model generalize to real-world conditions. We used 8631 identities with 2821697 total images from VGGFace2 dataset to train the face recognition model. The LFW dataset was used as a hold-out set due to past work using this as a benchmark, which allows our system to more easily be compared to other face recognition system. In total, we used 5749 identities with 13172 total images for testing.

Finding the facial region in the image is implemented with the help of the open-source library Dlib[5]. This library provides the fast face alignment algorithm proposed by Kazemi and Sullivan [7]. We utilized the Dlib face detector for face cropping on both the VGGFace2 and LFW datasets as a pre-processing step. This step both cropped and aligned the images such that they were 96×96 and transformed facial features to be in consistent positions within these images. The Dlib face detector has an accuracy of 99.54% on the LFW dataset when utilized to detect facial regions.

4.2 Training

In all our experiments, we trained our neural network with the VGGFace2 dataset with the stochastic gradient descent method. Hyperparameters are slightly modified from recommended values in [13]. Initial hyperparameter tuning focused on ensuring stable learning with the new VGGFace2 dataset and modified model.

All other training methods followed [13] in which we sampled identities and images for those identities from the dataset in order to produce triplets of the form (anchor, positive, negative). These triplets were used in the triplet loss in order to widen the margin between identities. In order to ensure fast convergence, given an x_i^a, we want to select an x_i^p such that $argmax_{x_i^p} ||f(x_i^a) - f(x_i^p)||_2^2$. Similarly, we want to select an x_i^n so that $argmin_{x_i^n} ||f(x_i^a) - f(x_i^n)||_2^2$. Triplets which already satisfy the inequality in Equation 1 do not contribute to the loss function in Equation 2. Therefore, these triplets will not contribute to training and waste computational resources. We found that bad local minima occur early in training if we select hard negatives. Thus, triplets were rejected from training if they satisfied the inequality in Equation 1 already or did not fall within the semi-hard negatives margin α. As defined in [13], semi-hard negatives are negative examplars x_i^n which satisfy the following inequality,

$$\left\|f(x_i^a) - f(x_i^p)\right\|_2^2 < \left\|f(x_i^a) - f(x_i^n)\right\|_2^2. \tag{3}$$

Algorithm 1 Algorithm for Generating Triplets

```
procedure GenerateTriplets(people)
    triplets ← ∅
    for each person in people do
        seen ← ∅
        for each anchor in person.images do
            seen ← seen ∪ {anchor}
            for each positive in person.images \ seen do
                negative ← sample a random image from all people except person
                if Eq. 1 is violated and Eq. 3 is satisfied then
                    triplets ← triplets ∪ {(anchor, positive, negative)}
    return triplets
```

For every batch we sampled 720 people and 5 images per person. These images and identities were then used to produce triplets with the algorithm given in Algorithm 1.

These remaining triplets were then provided to the model as inputs utilizing mini-batches of size 90. The images for each triplet (anchor, positive, negative) were each fed through the model to produce an output vector. The mini-batch triplet loss shown in Equation 2 was then computed using these vectors and the gradient of that loss was used for optimization.

We start with a learning rate of 0.05. Our model was trained for 240 hours and stopped when it appeared to stagnate in performance improvements. The margin α is set to 0.2. Triplet loss training is known to easily collapse the model into a degenerate 0 vector for all identities if the gradients are too large, so we also performed gradient norm clipping. We trained the entire architecture with the AdaGrad optimizer [2] as it provided the highest stability in helping the model avoid collapse. The learning rate was exponentially decayed as training progressed in order to improve convergence of the model. Weight decay was also applied to constrain the magnitude of weights.

5 RESULT AND EVALUATION

We tested our client-server system structure by recording the time our system used to complete the face recognition task. We evaluated the whole system by making the face recognition request 50 times and recording the time span of Bluetooth transmission and the

[5] http://dlib.net/

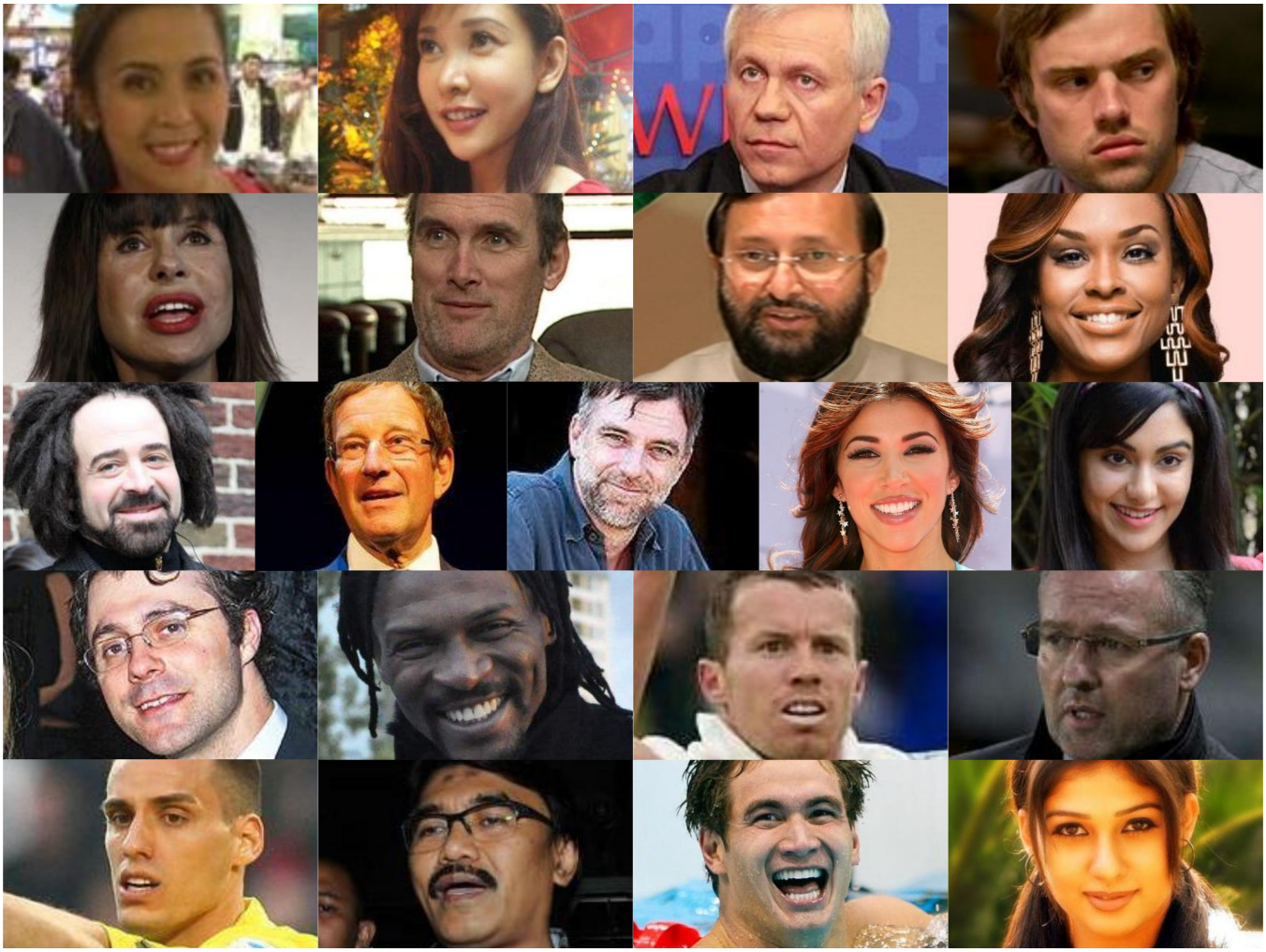


Figure 5: Collage of Random Faces from VGGFace2 Dataset

total time to receive the recognition result. We first recorded the time span between users pressing "send request" button and the captured image being displayed on the phone. Similarly, we also recorded the total time span between users making a request to take a picture and the recognition result being displayed on both the phone's and the glasses' screen. The average of 50 trials for both experiments are shown in table 1. The primary bottleneck is the transmission speed of the high resolution image between the devices. The system will scale efficiently as networks expand their bandwidth due to the primary limiting factor being the transfer speed. Ignoring transfer time, our model and user profile can detect and identify a person in an image within 200 milliseconds.

One of the most common cases to deploy our system in social events is when a user runs the recognition task while approaching a person of interest. From the average time needed for our system to complete the face recognition task, we can see that it is an acceptable time span for user to run face recognition before engaging in a face-to-face conversation. We also monitored the battery consumption when performing face recognition task. For most local-running systems, the battery life drops drastically within 100 recognition tasks. In our experiments, both devices only dropped less than 3% of the battery life.

Table 1: Performance Statistics for Face Recognition Task

Task	Average time(seconds)
Bluetooth transmission	5.348s
Total time for user to get result	13.69s

Figure 6 visualizes the centroid vector of different identities. We use t-SNE [17] to reduce the facial embedding from 128 to 2 dimensions and visualize it on a sample of identities from LFW. Different labeled identities are shown with different colors and centroid vectors are shown as darker shades of the same color.

We evaluate our back-end model on the LFW dataset on a face verification task. Namely, given a pair of two face images (i, j), our model compares the squared L_2-distance between their embedding

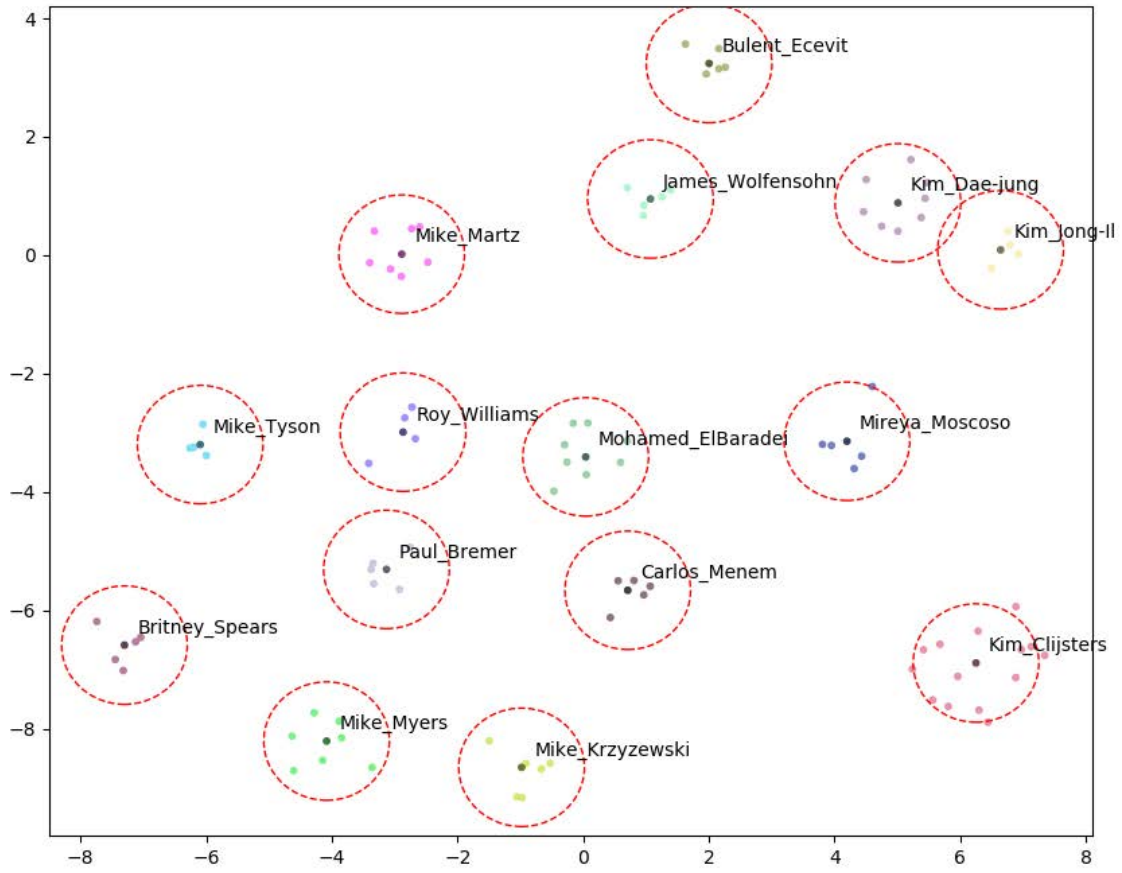


Figure 6: Facial embedding and centroid vectors for a sample of identities in LFW dataset

vectors. A threshold $D(i, j)$ is used to determine whether this 2 images belong to the same identity or not. All the images of same identity are denoted as ID_{same}, while all the images of different identities are denoted as ID_{diff}. We used both accuracy and validation rate to evaluate our model. Our definition of evaluation metrics are the same as those defined in [13].

The true accepts are the face pairs correctly classified as same faces at threshold d. It is defined as:

$$TA(d) = \{(i, j) \in ID_{same}, with\ D(i, j) \leq d\}. \tag{4}$$

The false accepts are the face pairs incorrectly classified as same faces at threshold d. It is defined as:

$$FA(d) = \{(i, j) \in ID_{diff}, with\ D(i, j) \leq d\} \tag{5}$$

The validation rate is defined as:

$$VAL(d) = \frac{|TA(d)|}{|ID_{same}|} \tag{6}$$

Similarly, the false accept rate is defined as:

$$FAR(d) = \frac{|FA(d)|}{|ID_{diff}|} \tag{7}$$

We follow the standard protocol for *unrestricted, labeled outside data*. 10-fold cross validation was used to evaluate our model on the LFW dataset. Nine training splits are used to select the squared L_2-distance threshold. The tenth split is used to perform classification. By 10-fold cross validation, our model produced one threshold for a final accuracy as well as one threshold for a final validation rate. Final accuracy on the LFW test set is 98.18% with an optimal threshold of 1.219. Final validation rate is 88.85% with a false accept rate of 0.001 and an optimal threshold of 0.923. The threshold for

the final validation rate is also used in further experiments as a rejection threshold for unknown identities. For any new face image, we determine if the identity is unknown if the minimum squared L_2-distance between its embedding and the centroid vectors of all other identities is larger than 0.923.

6 CONCLUSION AND FUTURE RESEARCH

The project's objective is to produce a real-time facial detection and recognition system which can run on mobile or wearable glasses to provide extra information in social events, especially for individuals who have prosopagnosia.

Such systems have been proposed during past couple years. However, constraints were imposed (re-training needed for new identities, large battery consumption). Our system combines modern computer vision machine learning models with recent wearable technologies. Two applications both on a pair of smart glasses and a cellphone as well as a deep convolutional neural network running on a separate computer were developed. Good results on the accuracy of the face recognition task was obtained with the approach we investigated in our system. We showed that our model is able to adjust the identity clusters based on new face embedding produced by the neural network. Thus, no rebuilding process is needed for new identities. According to our result of the computation efficiency experiment, it is possible for our system to provide identity information in real-world social events.

In the future, we will be working on optimizing the client applications for better user experience and higher computation efficiency. We will experiment with alternate face detectors which may perform better in real-world conditions. Additionally, we plan to invite more people to carry out further experiments over a longer period of time with more content in order to measure the effects on real social events. We will also test different back-end models, such as newer convolutional architectures like DenseNet which makes more use of residual connections [5]. We will also train larger models on more powerful machines to improve the performance of face recognition tasks by processing more details in face images.

REFERENCES

[1] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and ANdrew Zisserman. 2018. Vggface2: A dataset for recognising faces across pose and age. In *2018 13th IEEE International Conference on Automatic Face Gesture Recognition (FG 2018)(FG)*, Vol. 00. IEEE, 67–74. https://doi.org/10.1109/FG.2018.00020

[2] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12, Jul (2011), 2121–2159.

[3] Thomas Grüter, Martina Grüter, and Claus-Christian Carbon. 2008. Neural and genetic foundations of face recognition and prosopagnosia. *Journal of Neuropsychology* 2, 1 (2008), 79–97. https://doi.org/10.1348/174866407X231001

[4] Kaiming He, Xiangyu Zhang, Weidi Xie, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 770–778.

[5] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[6] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. 2007. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments.* Technical Report 07-49. University of Massachusetts, Amherst.

[7] Vahid Kazemi and Josephine Sullivan. 2014. One millisecond face alignment with an ensemble of regression trees. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1867–1874.

[8] Ingo Kennerknecht, Thomas Grueter, Brigitte Welling, Sebastian Wentzek, Jürgen Horst, Steve Edwards, and Martina Grueter. 2006. First report of prevalence of non-syndromic hereditary prosopagnosia (HPA). *American Journal of Medical Genetics Part A* 140, 15 (2006), 1617–1622. https://doi.org/10.1002/ajmg.a.31343

[9] Sreekar Krishna, Greg Little, John Black, and Sethuraman Panchanathan. 2005. A wearable face recognition system for individuals with visual impairments. In *Proceedings of the 7th international ACM SIGACCESS conference on Computers and accessibility (Assets '05)*. ACM, New York, NY, USA, 106–113. https://doi.org/10.1145/1090785.1090806

[10] Guangcan Mai, Kai Cao, Pong C. YUEN, and Anil K. Jain. 2018. On the Reconstruction of Face Images from Deep Face Templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2018), 1–1. https://doi.org/10.1109/TPAMI.2018.2827389

[11] Bappaditya Mandal, Shue-Ching Chia, Liyuan Li, Vijay Chandrasekhar, Cheston Tan, and Joo-Hwee Lim. 2014. A wearable face recognition system on google glass for assisting social interactions. In *Computer Vision - ACCV 2014 Workshops*. Springer, Springer International Publishing, 419–433.

[12] Davide Rivolta. 2014. *Prosopagnosia: The Inability to Recognize Faces*. Springer Berlin Heidelberg, Berlin, Heidelberg, 41–68. https://doi.org/10.1007/978-3-642-40784-0_3

[13] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. FaceNet: A Unified Embedding for Face Recognition and Clustering. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 815–823.

[14] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning.. In *AAAI*, Vol. 4. 12.

[15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going Deeper With Convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[16] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. 2014. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 1701–1708. https://doi.org/10.1109/CVPR.2014.220

[17] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

[18] Xi Wang, Xi Zhao, Varun Prakash, Weidong Shi, and Omprakash Gnawali. 2013. Computerized-eyewear based face recognition system for improving social lives of prosopagnosics. In *Proceedings of the 7th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth '13)*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium, 77–80. https://doi.org/10.4108/icst.pervasivehealth.2013.252119

[19] Wenyi Zhao, Rama Chellappa, P. Jonathon Phillips, and Azriel Rosenfeld. 2003. Face recognition: A literature survey. *ACM computing surveys (CSUR)* 35 (2003).