Real-Time Classification of Unknown Faces

Cameron Doggett
Email: camd.99@utexas.edu

Pranav Akinepalli
Email: pa8789@utexas.edu

Steven Sun
Email: stevensun914@utexas.edu

I. Abstract

We have designed a facial recognition program capable of learning the features of previously unknown individuals in near real-time from a live camera feed. To accomplish this the program combines various well-established technologies such as a histogram of oriented gradients for facial detection, a deep neural network for extraction of the features of unknown faces, and a linear support vector machine to classify the faces when shown again.

We evaluated our system by sampling from the student population at the University of Texas at Austin. Our results for ten different subjects was a 100% classification accuracy with a sample mean confidence of .9032 and a standard deviation of .0433. These results are very promising for future extension towards real world application.

II. Introduction

A. Problem

The main problem that we wanted to tackle was the consistent recognition of faces, not just ones that the system had been trained on, but also faces that it had never seen before. Facial recognition in general is a very useful feature and we figured that implementing software to do so, especially software that was flexible enough to be implemented on systems such as the Building-Wide Intelligence robots, would be a great goal to strive towards. Facial recognition opens up all sorts of possibilities including perhaps performing a protocol when a person is encountered and either registering said person if they have never been encountered or greeting them if they have.

B. Solution

The HOG face detector was used to find and align facial landmarks in specific positions for forwarding to the deep neural network. Once the features were extracted, they were stored as a representation vector and sent to the linear support vector machine where the respective face was classified depending on whether the system had already seen the face before. If the classifier identified the face as unknown, the classifier was retrained to include that face. The system was then evaluated by its ability to analyze a frame from video feed and classify any faces it found as unknown if the face was not previously assigned a label. From there, the face was labeled and presented to the system at various angles, after which we recorded the confidence with which each face was classified.

III. Background

Two main analytical approaches have been used regarding facial recognition, most commonly holistic methods versus feature based methods.

Holistic methods focus on having a face as an input and creating it in a lower dimension. This entails such techniques as principal component analysis. The basic principle rests on using standardized faces and describing any new input as a combination of those standardized faces. The standardized faces used most often are eigenfaces. The greatest benefit with this approach is that the full facial input is used to create a relationship between the input face and the eigenfaces. However, new eigenfaces need to be made for varying orientations and positions so this scales exponentially in terms of the increasing number of factors taken into account when recognizing a face. Thus, it can prove to be more difficult to achieve a sort of general classification when using this method. Another more computationally heavy method has been developed called auto-encoding for facial recognition. This method is able to learn non-linear feature representations which principal component analysis cannot do. It has been shown that auto-encoders perform better than principal component analysis, especially when given large inputs [1].

Feature based methods represent the more innate approach of attempting to pick out certain features that can be used to distinguish the given input face from others. Since a combination of these features is used to classify a face, this approach is much better at variations in the natural face than the holistic approach is. However, it is much harder to enforce the features that constitute the general structure of the face. Thus, determining what features to use, how many to look for, and the importance of each opens up the possibilities of many representations for even just one face alone and can make it harder to search for a given face [2]. Feature based recognition is the approach we decided to take because it is more intuitive in nature compared to other methods.

IV. Approach

A. Alignment

Alignment is the process of producing a cropped image containing a person's face, with features placed in standardized, fixed locations for later feature extraction.

1) Face Location Detection: Facial detection is the first step toward producing an aligned image. There are several libraries across multiple languages capable of facial detection, however, we found the face_recognition and OpenFace packages for

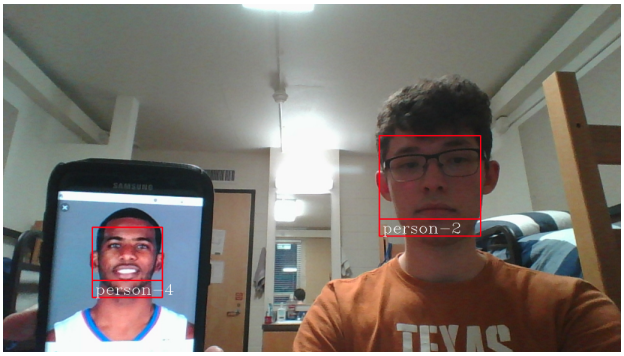

Fig. 1. Single processed frame with bounding boxes and labels placed over two learned faces.

Python to have the best documentation and largest development communities surrounding them [3]. While OpenFace and face_recognition provide much of the same functionality in terms of facial detection, we decided on face_recognition's detection because we desired as much flexibility as possible and it supports two methods of facial detection, one being a histogram of oriented gradients and the other a more accurate deep neural network. We added support for both HOG and DNN detection in our program, however it should be noted that the latter is computationally expensive and requires a CUDA compatible GPU in order to run in reasonable time. For our purposes, we only used HOG because we were limited to a slightly dated laptop with no dedicated GPU. Also, in order to improve performance, frames are downscaled to half of their original size and converted to grayscale before being passed to HOG.

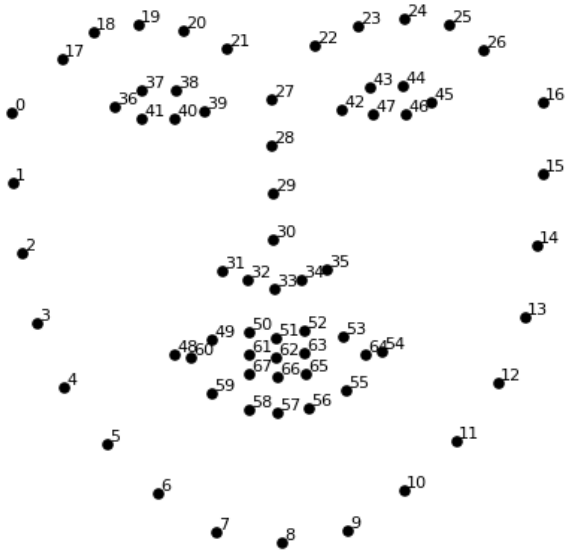

Fig. 2. Figure depicting the 68 point model used for landmark detection.

2) Landmark Detection: After obtaining the face bounding boxes for a frame, landmark detection must be run over the detected face locations in order to determine where certain features of a person's face are in an image. This is necessary for obtaining standardized results later from the feature extractor. Landmark detection was accomplished using a 68 point landmark model (see figure 2) and the real-time pose estimator from dlib [4]. Points in the model correspond to the relative locations of the mouth, chin, nostrils, eyes, and more. To complete the alignment process and produce a cropped image, we used OpenCV's affine transform function to create a square image containing only the person's face and facial landmarks [5].

B. Feature Extraction

After the faces in a frame have been aligned, they are then passed to a neural network for feature extraction. We chose the deep neural network implemented with Torch from OpenFace for feature extraction since the produced embedding accounts for Euclidean distance which produces better classification results [3]. Like the aforementioned deep neural network for facial detection, this DNN is also slow on CPUs compared to CUDA enabled GPUs, however this is compensated for in the actual program by processing every fifth frame instead of every frame. The representations returned for each face can then be easily written to a database or file along with an associated label. The files containing the representations and labels can then be read each time the classifier must be retrained, meaning the original frame does not need to be saved.

C. Classification

The representation gathered from the feature extractor can then be used to train a classifier which will be capable of classifying future representations. To establish a base classifier for the webcam application, more than 400 images, most of which are from the Caltech Faces dataset, of several different people were labeled as unknown and trained on. This was done in order to establish the unknown class as a broad label so that any unlearned faces will be classified as unknown rather than an already learned person. For the classifier model, we selected the support vector machine with a linear kernel from the sklearn package because of its relatively fast training speed compared to the other models [6]. When guessing the labels of faces from incoming frames, instead of training on a new face as soon as its labelled as unknown once, the program instead waits for that face to be labelled unknown at least 5 consecutive times before saving 30 representations of the person from different frames and retraining after. This is done in order to ensure that a person classified as unknown is actually not apart of the learned set of faces rather than a result of error.

V. Evaluation

In order to evaluate the performance of our real-time facial classification technique on previously unknown individuals, we tested how well our program can classify previously unidentified individuals as unknown and then subsequently

Subject	Classified as unknown	Mean confidence
Male		
1	Yes	0.9273
2	Yes	0.9466
3	Yes	0.8486
4	Yes	0.8609
5	Yes	0.8814
Female		
6	Yes	0.9696
7	Yes	0.9544
8	Yes	0.8774
9	Yes	0.8645
10	Yes	0.9016

TABLE I
RESULTS FOR CORRECT CLASSIFICATION AS AN UNKNOWN FACE AND SUBSEQUENT MEAN CONFIDENCE OF FURTHER CLASSIFICATION AFTER ADDING FACE AND LABEL TO REPRESENTATION SET.

learn that person's particular facial features followed by classifying them with a unique label. First, we needed to evaluate whether or not our program can consistently classify each student participant as unknown initially, successfully retrain the classifier with their features accounted for, and afterward consistently classify the same student with a label unique only to them. Second, we tested the effectiveness of the retraining process by having the participant move around in the frame such that they were not directly in front of the camera. Data collected from the first and second test will demonstrate the effectiveness of our program.

VI. RESULTS

Across different times, 10 people were individually filmed in real time to test our facial classification process. Starting with the placement of Caltech Faces, an image set of 450 frontal pictures of 27 people, into an "unknown" classification label, each face was subsequently introduced to the camera for approximately half a minute. The subjects changed facial angle relative to the camera only slightly but moved around a decent amount. Representations for each face were kept across tests - for example, by the time subject 10 was tested, representations for the previous 9 subjects were already saved in the program. For these people, every single one was correctly classified as unknown. Following unknown classification, a new label was created and further scans of the same face would return a match to the new label with given confidence (Table 1). For around 30 seconds of video input for each face, the mean confidence was found by summing all confidences retrieved during said time period and dividing by number of faces processed.

The "hit rate" of detecting new faces and properly labeling them after updating the representation set was 1.0 for this small sample size of 10. Furthermore, the sample mean confidence was .9032 with a standard deviation of .0433. These results indicate strongly reliable classifications using minimal training data and thus deliver hope towards extending this classification software to real world applications.

VII. DISCUSSION

Though our testing sample was small due to logistical constraints, the results were overwhelmingly positive, indicating that this method of face recognition is viable at least for small groups of people. Performance with a greater amount of people and stored representations may become an issue, though realistically any application of this technique wouldn't require more than a few thousand unique faces to be saved due to the upper limits of people that can fit a space with an operating service robot. The retraining process is the only task affected by increasing the training set, and for a few dozen people, re-training completed on the order of milliseconds. By extension, since the only time retraining needs to occur is when a new person of interest is found, the percentage of time overall spent retraining will be minimal. The majority of time spent was during decomposition and storing representations, which would always run in constant time in relation to how many representations were previously stored.

A large gap between practical application and our current build lies in densely populated environments. The identification and retraining process all require faces to be angled within a certain threshold of the camera. Additionally, when retraining, the ideal photos to extract representations from involve faces that face relatively straight on towards the camera. Problems would likely arise when training data consists of near-profile shots of faces (see figure 3). However, given faces looking near the camera, facial detection and classification can occur even with multiple faces in one shot. Given that it is unlikely for a crowd of people to all face a camera at once, modifications to our code would be required to reliably remember people simply passing by the robot.

Further, when an unknown face is detected, only one can be on screen while re-training occurs. Since there is no way to label and identify which face needs to be stored in representations aside from a positional index, training with multiple faces on screen is unreliable. Due to the delay involved with a live camera feed and all the operations run per frame, there is no guarantee that the position of each person can be tracked during the training process. If a person next to the unknown target was already labeled, the position of the unknown person could be located by first finding their labeled neighbor. In the

Fig. 3. HOG is unable to detect faces which are in profile or angled.

case of both neighbors being unknown, this approach wouldn't work. For the case of a labeled neighbor, efforts could be made towards improvements in hardware and optimizations in code. However, our current build doesn't run fast enough to operate in this labeled neighbor case. Additionally, different facial detection algorithms were explored to speed up run time. Primarily, dlib's facial detection is one of the fastest, but it has a glaring flaw in that its face detection frequently cuts out portions of the face such as the chin. In addition, dlib doesn't detect any faces smaller than 80x80 pixels. A compromise would have been OpenCV's facial location algorithm which is only slightly slower than dlib. Unfortunately, OpenCV's facial detection was incompatible with OpenFace, which was required for classification.

Overall, our facial classification system performs very well in an environment where people are conscious of the need to have their face initially be seen and classified in an individual setting. For smaller groups of people, it is very feasible for the entire group to be trained into the data set through short interactions with a robot running our software. These do not need to be pre-planned sessions but more along the lines of first-time interactions where the user consciously looks towards the camera for a few seconds before continuing what they were doing in a more natural manner. Following, applications such as voice to face association can be integrated to perform tasks such as delivering drinks to people who have asked for it off-camera.

VIII. Conclusion

We performed an evaluation of our facial recognition software by first presenting each of the ten subjects' faces individually to the camera to see if our software would be able to recognize whether or not we were showing it an unknown face. We found that it was able to consistently recognize that it was being shown a person which it had never seen before. Once we had done that for each of the faces, we took each face individually and presented it to the camera at a variety of angles for it to identify the face from. The mean confidence of each face was determined by calculating the amount of total confidence the software yielded for every image of a face the software processed divided by the total number of images of one face that were processed. This would result in the average confidence the software had in identifying a particular face. All 5 males yielded an average confidence 0.89296 while the 5 females yielded an average confidence of 0.9135. This resulted in a total average confidence of 0.90323.

On average, our system was about 90% confident in identifying the ten faces given to it and it was 100% confident in initially recognizing that it was being given an unknown face. From our results, we saw that the females were consistently higher in terms of confidence compared to the males. This could lead us to believe that females tend to have more distinctive faces compared to males and when training on a set that is limited in size, it may be important to train in more males than females for better results. Looking at our results, the fact that the lowest confidence was only 85% and the highest was up to 96% means that our software was rather consistent in identifying previously unknown faces on the fly with a high enough confidence to boot. Finally, our rather high average confidence of 90% per face means that our facial recognition software does a pretty good job of identifying faces, especially those that were never encountered before. Factoring this in with the fact that, most of the time, faces will be identified when they are looking directly at the camera, we can confidently say that our facial recognition software will usually be spot-on. This leaves a flexible and accurate facial recognition software which could serve a variety of purposes, including the purpose of serving as prime facial recognition software for the Building-Wide Intelligence robots.

References

[1] K. Siwek and S. Osowski, "Autoencoder versus pca in face recognition," 2017.

[2] M. Stollenga, "Using guided autoencoders on face recognition," 2011, p. 13.

[3] B. Amos, B. Ludwiczuk, and M. Satyanarayanan, "Openface: A general-purpose face recognition library with mobile applications," CMU-CS-16-118, CMU School of Computer Science, Tech. Rep., 2016.

[4] D. E. King, "Dlib-ml: A machine learning toolkit," *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.

[5] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

Acknowledgments

Thank you Dr. Hart and mentors for providing us with guidance during this research project and over the semester.