Smart face detection with 2FA on edge devices

Angad Sharma (17BCE2009)

Ubaid Usmani (17BCE0983)

Vellore Institute of Technology, Vellore

ABSTRACT

There are a lot of available algorithms for face detection system, but in sectors where legacy software is kept unmaintained for decades, a more robust system needs to be in place on facial recognition systems.

Edge devices can be used to run advanced algorithms on sensory data and authenticate/identify users in these sectors. The code can simply be programmed on the edge devices and can be forgotten about. Our aim is to design such a robust system with the help of face recognition and two factor authentication.

10th September, 2020

Table of Contents

Abstract 0
Table of contents 1
Introduction 2
Literary survey 3
Comparison of biometric traits 4
Problem definition 7
Objectives 8
Proposed architecture 9
Implementation 11
Result analysis 15
References 16

1. Introduction

Face recognition systems are extensively used in both government as well as non government sectors. They have been extensively adopted by the smartphone industry as a means of authentication users, choosing speed over security on authentication. But in systems where identity disclosure is a risk and where false positives lead to the potential risk of leaking sensitive data, more robust system needs to be provided.

Consider customs, or banking sector for instance. They have mountains of legacy code running on their servers which is hardly maintained, especially in third world governments. Servers can also be a potential target of hackers as they provide a central source of failure for the entire system, even with extensive load balancing. To mitigate the risks of a central point of failure, edge devices are being adopted. The benefit being a veritable grid of information which is being processed on the client edge device and then synced up to a server. This reduces the risk of exposure and especially the risk of failure.

Edge devices nowdays are equipped with potentially usefull sensors and actuators which can record biometric data as well as enough compute to process the said data and produce meaningful information for the server to digest. Take the new Raspberry pi (2020) for instance. The highest tier of this device is equipped with 8 GB of random access memory along with a CPU which can be overclocked to around 2.1 GHz. Being a relatively cheaper alternative to a full desktop compute, it is but a credit card size piece of equipment with has a fully equipped ARM v8 instruction set and a lot of open source support. Such a computer can be used for executing facial recognition algorithms in legacy use cases.

Face recognition has proved to be a difficult challenge to solve over the years, with more research leading to the development of newer and more accurate algorithms for acheiving the same. In high-sensitivity legacy systems, facial recognition alone cannot (and should not) be the only means of authentication and authorization, hereby generating a need for another mechanism to go along with it, which is multi-factor authentication (MFA). MFA utilizes something that a user has to authenticate a user. For example: a one time password sent to the user as a message could function as a valid MFA method. In this paper, let us see how we can devise our own edge compute based MFA solution to add robustness to legacy facial recognition systems.

2. Literary Survey

Face recognition has many different Implementations over the years. Convulutional neural networks are generally used, but a very interesting implementation was the combination of Gabor Filters with CNNs for face detection Duch, 2005 and (Wu, 2002) which also take the presence of glasses as another source of variations we have to take into account, as opposed to the canelable biometric filtersSavvides, 2004. The first stage uses the Gabor filter which extracts intrinsic facial features. As a result of this transformation we obtain four subimages. The second stage of the method concerns the application of the convolutional neural network to these four images. The approach presented in this paper yields better classification performance in comparison to the results obtained by the convolutional neural network alone.

In addition to this Bayesian face recognition has shown efficient results (Moghaddam,) Bayesian face recognition is a technique for direct visual matching of images for the purposes of face recognition and image retrieval, using a probabilistic measure of similarity, based primarily on a Bayesian (MAP) analysis of image differences. The performance advantage of this probabilistic matching technique over standard Euclidean nearest-neighbor eigenface matching was demonstrated using results from DARPA's 1996 "FERET" face recognition competitionPhillips, 2000, in which this Bayesian matching alogrithm was found to be the top performer. In addition, we derive a simple method of replacing costly computation of nonlinear (on-line) Bayesian similarity measures by inexpensive linear (off-line) subspace projections and simple Euclidean norms, thus resulting in a significant computational speed-up for implementation with very large databases.

Then there is a technique which directly learns mapping from face images to a compact Euclidean space, Facenet (Schroff, 2015) Once this space has been produced, tasks such as face recognition, verification and clustering can be easily implemented using standard techniques with FaceNet embeddings as feature vectors. The method uses a deep convolutional network trained to directly optimize the embedding itself, rather than an intermediate bottleneck layer as in previous approaches. To train, we use triplets of roughly aligned matching / non-matching face patches generated using a novel online triplet mining method. The benefit of the approach is much greater representational efficiency: we achieve state-of-the-art face recognition performance using only 128 bytes per face. On the widely used Labeled Faces in the Wild (LFW) dataset, the system achieves a new record accuracy of 99.63%. On YouTube Faces DB it achieves 95.12%. Our system cuts the error rate in comparison to the best published result [DeepId2+] by 30% on both datasets.

Despite significant recent advances in the field of face recognition [DeepFace, DeepId2]Wen, 2016, another interesting approach is using eigen faces (Alex, 2000) , which we have done in this project. This approach treats face recognition as a two-dimensional recognition problem as opposed to laplacian facesHe, 2005 , taking advantage of the fact that faces are are normally upright and thus may be described by a small set of 2-D characteristic views. Face images are projected onto a feature space ("face space") that best encodes the variation among known face images. The face space is defined by the eigenfaces, which are the eigenvectors of the set of faces; they do not necessarily correspond to isolated features such as eyes, ears, and noses. The framework provides the ability to learn to recognize new faces in an unsupervised manner.

We have also seen the convergence of deep learning and edge computingWang, 2020 with research going as early as the second quarter of 2020. Our approach will be combining the eigenfaces algorithm for facial recognition with the implementation of a deep learning hub in an edge computing device, namely a raspberry pi 3 model B+.

3. Comparison of face recognition algorithms

Facial Recognition Algorithms			
Algorithms	**Description**	**Pros**	**Cons**
Eigenface-based method	**Efficiently representing faces using PCA (Principal Component Analysis). Their goal of this approach is to represent a face as a coordinate system. The vectors that make up this coordinate system were referred to as eigenpictures.**	**> Easy Approach > Efficient Storage and processing time > Manages Dimensions by converting them to a latent space**	**> Sensitive to lightning and position of head. > Time Consuming in generating eigenvectors**
Neural Networks	**Many pattern recognition problems like object recognition, character recognition, etc. have been faced successfully by neural networks. These systems can be used in face detection in different ways.**	**> Best feature extractor for images because of SOTA CNN models. > Better results and give robust results than eigen faces and graph matching algorithms.**	**> Highly complex algorithm tough to understand. > Training time and resource consumption is too high > Prone to trojan and spoof attack.**
Fisherface algorithm	**Fishers Linear Discriminant (often FLD and LDA are used interchangeably). Its closely related to PCA. FLD attempts to model the difference between the classes of data, and can be used to minimize the mean square error or the mean absolute error.**	**> Within class information is used to minimize variation in the same class. > Lightening and head position can be tackled with this method > Similar to eigenfaces but with better classification enhancement.**	**> Error rate is high > More complex than eigenfaces for finding the projection of faces. > Large storage required and processing time is high in recognition.**

Facial Recognition Algorithms			
Algorithms	**Description**	**Pros**	**Cons**
Elastic Bunch Graph Matching	**Each subject has a bunch graph for each of its possible poses. Facial features are extracted from the test image to form an image graph. This image graph can be compared to the model graphs, matching the right class.**	**> Changing or missing any one feature it does not mean that the person will not recognized. > No extra effort for adding new image to the database > Possible to recognize a person upto rotation of 22 degrees.**	**> Very sensitive to lightning conditions > Graphs have to be put manually on the face. > Recognition rates decrease significantly when changes in lighting are large.**
Support Vector Machine (SVM)	**It is a new type of pattern classifier which is based on novel statistical learning techniques. SVM works well with high dimensional spaces under small training samples. It gives better results than traditional.**	**> Uses Structural risk minimization > SVM cannot suffer from their theoretical weakness. > The development of SVMs involved sound theory first, then implementation and experiments.**	**> The performance of SVM depends upon the kernel we chose. > The size is the problem both in training and testing. > SVM is slower than Neural Network.**
KanadeLucasTomasi feature tracker	**The Kanade Lucas Tomasi Feature tracker in computer vision is used to feature extraction. KLT uses spatial intensity information which directs the search for position which gives the best match. It works better than traditional techniques as it gives more accurate results than traditional.**	**> KLT works better for textured pixels. > KLT is much quicker than other methods for checking lesser probable matches between pictures. > KLT finds a good point to track from frame to frame.**	**> KLT does not hold brightness constancy. > KLT gives error when motion is large, so to fix it we have to match key points. > Small errors occur when the appearance model is updated.**

Facial Recognition Algorithms			
Algorithms	**Description**	**Pros**	**Cons**
Independent Component Analysis (ICA)	**Independent component analysis is a method to find factors or components from multi dimensional statistical data. There is a need to implement a face recognition system using ICA for facial images having face directions and different lighting conditions, which will give better results as compared with existing systems.**	**> ICA provided a more powerful data representation than PCA. > PCA_ICA attains higher average success rate than Eigenfaces, the Fisher face and methods**	**> The ICA model equation one cannot determine the variances of the independent components. > Cannot rank the order of dominant components.**
Local Binary Pattern (LBP)	**LBP is the best performing texture descriptors and widely used in various applications. It has proved itself to be highly discriminative and because its invariance to monotonic gray level changes and computational productivity, make it suitable for demanding image analysis tasks. Face can be seen as a composition of micro-patterns which can be well described by LBP operators.**	**> LBP tolerance to monotonic gray-scale changes. > The recognition rates of the LBP maintain high level under the effect of localization errors**	**> The recognition rate of the local region based methods is lower than that of PCA. > The binary data produced by LBP are sensitive to noise. > LBP produces long histograms, which slow down the recognition speed especially on large-scale face databases.**

4. Problem Definition

4.1. Problem Domain

The domain of the problem that we are going to solve is in the sectors which have legacy code bases and there is a need for on-premise facial recognition with zero fault tolerance. In lieu of our requirements, some of the beneficiaries of this project will be seen in the following sector:

Banking sector

Customs

Aadhar Card (Social Security) centers

University digital credit systems

Passport and VISA procedures

Non banking finance sector

Online taxi/cab services

4.2. Problem Description

On-premise facial recognition is a challenge in sectors where there is no fault tolerance, and where there is a single point of failure on an infrastructure level. Impersonation is a major issue and OTP alone is not enough for recognizing an individual since social engineering attacks lead to phone numbers and SIM cards being compromised. There is a need for a more robust solution.

Sectors such as the ones mentioned above have a function that requires users to be present for verification on premise. In such a case, mounted cameras are used for facial recognition where each camera is linked to a computer connected to a server, which aggravates the single point of failure. There is a need for a more robust alternative, which will be discussed in the next section.

4.3. Gap Analysis

As mentioned above, the following gaps exist in pre-existing facial recognition systems for legacy software within the problem domain for which we aim to build a solution.

Lack of easy to use on-premise MFA mechanisms. People are required to carry identification, and a lot of workplace hours are wasted verifying the documents for each person.

Centrality of servers is a major issue since all of the processing happens in a remote server. Many times this leads to downtime, especially in banks.

Old legacy systems use old facial recognition software which requires users to take off their spectacles. They often do not work when people wear coloured contact lenses.

As the legacy systems age, the users also age. The use of sub-par facial recognition algorithms lead to the need of re-uploading and catalogging people over time, which wastes a lot of office hours.

5. Objectives

Note that we are looking at an on-premise solution for the facial recognition problem along with avoiding a single point of failure, to acheive the same the following are some of the solutions that we aim to implement in this project:

Setting up a facial recognition center using Eigenfaces

Configure two factor authentication using TOTPs (time based OTPs)

Deploy the computation into an edge device (namely raspberry pi)

Operate the edge device remotely and gain metrics

Setting up an easy to use web-application for image acquizition

Setting up training data for recognizing and identifying faces

Minimizing false positives in the facial recognition approach

Syncing TOTPs to the users' own account

6. Proposed Architecture

6.1. Face Recognition Algorithm

The input of a face recognition system is always an image or video stream. The output is an identification or verification of the subject or subjects that appear in the image or video. Some approaches define a face recognition system as a three step process. From this point of view, the Face Detection and Feature Extraction phases could run simultaneously.

The first module that we will be using is face recognition using the eigenface algorithm. This is going to be done in our edge computing device

This algorithm is meant to run on the rapberry pi and base its data off of a provided data store replicated on the edge device as well. For all intents and purposes, the database can be a hosted one as well, but our aim is to limit the number of active central server calls as much as we can so that we only use it for cold storage and long term analysis.

The database of identities stored on the edge devices can be gergraphically divided on the basis of the location of the premise of use for the particular algorithm. For example, a branch of the Royal Bank of Scotland in Britain should only have the identities of the people living in the same area code as the branch.

6.2. Project Workflow

In this section we will be looking at the individual components of our solution and how they interact to each other in a production setting. The following are the modules involved in our implementation.

Edge Computing Device : Raspberry Pi Model 3 B+. This version has 1 GB of random access memory, 4 single threaded virtual CPUs and variable storage (which depends on an external SD card, which in our case is going to be a 64 GB variant. This device will run the ARM v7 32 bit Raspbian operating system under the hood, and has two interfaces open for access, namely SSH and HTTP(S).

Cloud TOTP Server : A server which is responsible for issuing and revoking OTPs. This server issues a token to the client directly, who then submits that particular token to our edge computing device, which queries this server again for token verification and validation. A fixed time is allotted for the latter, after which both the token as well as the deadline resets for that particular session and the user has to make another request for two factor authentication.

Cloud Sync Server : A server which syncs logs from edge computing devices for cold storage and long term analysis. Note that this is the central server that was mentioned before as a central point of failure, but in our implementation, this server is not responsible for real-time facial recognition and identification tasks, but more so as a log storage server for monitoring and analysis. These logs can be warehoused and exported for long term analysis.

Sensor Module : In this case it is a camera responsible for capturing high definition images. This camera can be attached to the edge device directly, and snap photographs in either real-time or in quick succession. Note that taking multiple photos is ideal for our algorithm since we can correlate

them for better results.

Face Recognition Module : This is the face recognition algorithm that runs at the edge computing device level and recognizes the names from a given data store after extracting the face from the provided image. The following diagram captures the workflow in its entirety:

Consider the following diagram for the full workflow explanation. Here, the client is actually an HTTP(S) service running on our edge computing device on a private network. Having a private network (i.e access to this device is not possible through the external network) is due to safety concern and the fact that the users of this web service will be present on premises to capture a real-time feed of the person of interest.

7. Implementation

7.1. Edge Computing Device Specification

The edge computing device that we have in question is a Raspberry Pi Model 3B+. This device has been chosen because of its abaility to run a full fledged ARM based linux distribution as well as the ability to be seamlessly connected with a camera. The following are the stats of our edge device:

1 GB RAM
Raspbian OS
ARM v7 Instruction Set
32 Bit Architecture
2 virtual CPUs

1 Left most: Jumper Cables (female to female)
2 Left top: I2C Module for Display
3 Center: Raspberry Pi
4 Right most: Jumper Cables (Male to female)
5 Top right: 16X2 Character Display
6 Right to the pi: Camera Module

7.2. Setting up the edge computing device

The first step is to take an SD card and plug it into your laptop. Then download the raspbian image and flash it into the card using the **dd** command.

sudo dd if=./image.iso of=/dev/<sd-card-name>

We will enable SSH so that we can connect to the raspberry pi. Run the **sudo raspi-config** command to enable SSH. Subsequently, we need to run the following command to set up SSH key exchange:

ssh-keygen -t rsa

Then run the following commands on the host machine:

scp pi@<ip-of-pi>:~/.ssh/id_rsa .

ssh -i ./id_rsa pi@<ip-of-pi>

7.3. Running the program on our device

Our program is available on GitHub. You need to clone it using **git** and then install dependancies to run it. Note that you need **python 3** installed on your raspberry pi for this project to run. This then opens up a web interface for you to capture images. Connect the camera module to the raspberry pi and make sure that the camera light is working. Then go into the project source directory and run the following commands:

pip3 install -r requirements.txt

which will download all of the requirements needed to run our system. Then you can run the **app.py** using python3 like this:

python3 app.py.

Doing so will open a port (namely 8080) on your raspberry pi, which you can then use with **localhost:8080** on the browser and see your webpage loaded on the screen. This interface contains the image capture and upload functionality, subsequent to which you have to enter an OTP sent to you.

Use **Control + C** to exit your server and return to the command line again.

7.4. Understanding the implementation

7.4.1. Facial Recognition

Facial recognition is implemented using EigenFaces algorithm to recognize and manipulate faces from Python or from the command line with the world's simplest face recognition library.

Built using dlib's state-of-the-art face recognition built with deep learning. The model (Geitgey,) has an accuracy of 99.38% on the Labeled Faces in the Wild benchmark.

This also provides a simple face_recognition command line tool that lets you do face recognition on a folder of images.

Although the algorithm works well in most cases, it has the following caveats:

1 The face recognition model is trained on adults and does not work very well on children.
2 It tends to mix up children quite easy using the default comparison threshold of 0.6.
3 Accuracy may vary between ethnic groups. Please see this wiki page for more details.

The face recognition algorithm library can be understood by the following lines of python code:

import face_recognition

picture_of_me = face_recognition.load_image_file(me.jpg)

my_face_encoding = face_recognition.face_encodings(picture_of_me)[0]

unknown_picture = face_recognition.load_image_file(unknown.jpg)

unknown_face_encoding = face_recognition.face_encodings(unknown_picture)[0]

results = face_recognition.compare_faces([my_face_encoding], unknown_face_encoding)

If the result if True in this case then the face is successfully recognized, otherwise it is not. This is quite a simple way to recognize faces but there is a lot happening under the hood of the algorithm that has been abstracted from us.

Although this library has a very high accuracy, some caveats (mentioned above) require the need of two factor authentication for additional robustness.

7.4.2. Two Factor Authentication

Two factor authentication in our platform is handled by an open-source authentication and authorization server providing 2-factor authentication and single sign-on (SSO) for your applications via a web portal. It acts as a companion of reverse proxies like nginx, Traefik or HAProxy to let them know whether queries should pass through. Unauthenticated user are redirected to Authelia Sign-in portal instead.

This solution contains the following features:

Several kind of second factor:

Security Key (U2F) with Yubikey.

Time-based One-Time password with Google Authenticator.

Mobile Push Notifications with Duo.

Password reset with identity verification using email confirmation.

Single-factor only authentication method available.

Access restriction after too many authentication attempts.

Fine-grained access control per subdomain, user, resource and network.

Support of basic authentication for endpoints protected by single factor.

Highly available using a remote database and Redis as a highly available KV store.

Compatible with Kubernetes ingress-nginx controller out of the box.

The TOTP is available only for a given time slice, after which it expires and a new TOTP is issues in its place. The user is required to scan their faces and then add the TOTP which has been generated on the web interface of the authenticator module. This TOTP is verified on the edge device by a network egress request to the provider servers, so we do not need to save TOTP states on the edge devices, ensuring that the image to be recognized and the second factor authentication have isolated environments of verification.

8. Result Analysis

The following results can be derived from our study of facial recognition algorithms and our implementation:

1 The eigenfaces algorithm we are using is 99.38% accurate

2 A major caveat is that this model is trained on adults and hence does not work well on children. Considering our use case domain to be limited to the banking sector, where minor accounts are maintained by adults, this is a viable and acceptable caveat

3 Time based OTP mechanism ensures two factor authentication, even in the case where a user might damage their face and render the facial recognition algorithm futile

4 Running our algorithms and analyses on an edge computing device prevents a single point of failure and ensure proper distribution of data, especially if we shard data according to a viable parameter, such as demographic

References

Duch, 2005.
W Duch, Kacprzyk J., Oja E., Zadrony S. (eds), *Artificial Neural Networks: Biological Inspirations – ICANN 2005. ICANN 2005.*, Lecture Notes in Computer Science **3696**, Springer, Berlin, Heidelberg (2005).

Wu, 2002.
Haiyuan Wu, Y Yoshida and T Shioyama, "Object recognition supported by user interaction for service robots,," *ICPR* **1**, pp. 107-110, Quebec City, Quebec, Canada (2002).

Savvides, 2004.
M. Savvides, B. V. K. Vijaya Kumar and P. K. Khosla in *Cancelable biometric filters for face recognition*, Proceedings of the 17th International Conference on Pattern Recognition, pp. 922-925, ICPR (2004).

Moghaddam, .
Baback Moghaddam,Tony Jebarab,Alex Pentlandb Mitsubishi Electric Research Laboratory in *Bayesian face recognition*, USAA Massachusetts Institute of Technology, Cambridge, Massachusetts.

Phillips, 2000.
P. J. Phillips, Hyeonjoon Moon, S. A. Rizvi and P. J. Rauss, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**(10), pp. 1090-1104, IEEE (Oct 2000).

Schroff, 2015.
Florian Schroff, Dmitry Kalenichenko, James Philbin, *Pr oceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 815-823, IEEE (2015).

Wen, 2016.
Y Wen, Zhang K., Li Z., Qiao Y. (2016), *ECCV*, Lecture Notes in Computer Science, **9911**, Springer, Cham (2016).

Alex, 2000.
Pentland Vision and Modeling Group Matthew A. Turk and P. Alex, *The Media Laboratory Massachusetts Institute of Technology*, Massachustts (November 5, 2000).

He, 2005.
Xiaofei He, Shuicheng Yan, Yuxiao Hu, P. Niyogi and Hong-Jiang Zhang, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **27**(3), pp. 328-340, IEEE (March 2005).

Wang, 2020.
X. Wang, Y Han, V. C. M. Leung, D. Niyato, X. Yan and X. Chen, *IEEE Communications Surveys & Tutorials* **222**(2), pp. 869-904, IEEE (Second Quarter 2020).

Geitgey, .
Adam Geitgey, "https://github.com/ageitgey/face_recognition," *GitHub.*