Journal of Natural Sciences and Engineering, Vol. 2, (2020)
DOI number: 10.14706/JONSAE2020213

Biometrics Based Access Control System

Mujo Hadžimehanović[1], Dino Kečo[1], Demir Korać[1]
[1]International Burch University, Sarajevo, Bosnia and Herzegovina
mujo.hadzimehanovic@stu.ibu.edu.ba
dino.keco@ibu.edu.ba
demir.korac@stu.ibu.edu.ba

Abstract **– Access control includes attendance checking and intrusion prevention. It is used to protect property, employees and other assets of a company or institution. Since attendance checking and intrusion detection are important segments of many educational institutions and other businesses as well, it is important to make these processes faster, easier and as convenient as possible. Lots of institutions are suffering from unreliable attendance checking methods, so we have decided to use biometrics, more precisely face recognition to automate and improve this overall process. As part of this study the full system has been implemented for recognition of people. As an example of usage in an educational institutions multiple photos will be recorded during the class session, so that in case of students leave class after the first shot, they will be removed from the attendance sheet. All recognized people will be stored in Mongo database as an array of features and later read from database and processed by using Python script for face recognition. All educational institutions are going to have benefits from this study. Benefits would be improving attendance management and security.**

Keywords - attendance, face, images, recognition

1. Introduction

Technology is rapidly improving nowadays and everyday activities are adopting these improvements. Point is to automate these activities and not to lose time performing them. Attendance is a really important part in most organizations such as schools, faculties, companies etc. Today, attendance is performed in various ways. Best way to do it is biometrics. Biometrics is a bioengineering area which is an automated method for person recognition based on its physiological or behavioural characteristics. There are many biometric templates such as fingerprints, face, hand geometry, iris, voice or signature. System is going to use face biometric template because it is the fastest approach and requires no human intervention. This method is better than other biometric methods because these methods are time consuming. There are also lots of systems which are using RFID cards, location based attendance tracking systems, signature based etc. Negative things about these methods are that they can be faked. In RFID and location based systems employees are carrying RFID cards or GPS locators. So, other people can check instead of other employees. There are two main stages of face recognition process and they are face detection and face identification [1]. Recording employees' work hours and their activities, attendance of students in schools are really important components of every company or school. This process is maintained by using signature, fingerprint, iris, RFID or face recognition. System is going to use a camera which captures images of people entering the company or school building. Detected faces will be compared with pictures which are already in the database. If a person's picture is in the database attendance will be checked automatically. Otherwise, the security system is going to be informed that an unrecognized person entered the building. So, this system can also be used as an intrusion detection system.

2. Literature Review

Following section contains a presentation of all related work and their methods.

Since they have large-scale data with massive noisy labels, X. Wu, R. He, Z. Sun, and T. Tan used a Light CNN framework [2]. Their Light CNN architecture contains Max-Feature-Map to suppress low activation neurons in all layers. Their model was trained on Celeb-1M dataset. In order to handle noisy labeled images, they proposed a semantic bootstrapping method to automatically re-label training data via pre-trained deep networks. For training purposes they used five types of databases. First type is commonly used Labeled Faces in the Wild which consists of ~13,250 images of ~5,750 people. At VR@FAR=0 for Light CNN-29, they achieved 97.50%, while results from all other methods were lower than 70%. Next type of the database are collections of images extracted from Youtube videos which contain YouTube Celebrities (YTC) and Celebrity1000 database. Precision achieved for these datasets is 94.18%. Third type are MegaFace, IJB-A and IJB-B datasets which are challenging and they got 85.13% precision. Cross-domain databases are the fourth type of database. It includes CACD - VS, Multi - PIE and the CASIA NIR - VIS 2.0 database. They achieved 98.55% on CACD, 95.0% on Multi-PIE and on CASIA VR@FAR=0.1% result is further improved from 94.03% to 94.77%.

M. Arsenovic, S. Sladojevic, A. Anderla, and D. Stefanovic use Convolutional Neural Networks (CNNs) cascade to detect faces and CNN to generate embeddings of each face [3]. Fact is the best results for larger datasets are achieved by using CNNs, but in their production environment that was not the case. CNN gave the best results for smaller datasets. Accuracy of 95.02% was achieved on a dataset created by authors in the real-time environment. Five employees of the company took pictures of themselves and they used these pictures as a dataset. Model was trained with these 5 pictures.

Active annotation and learning framework was used by H. Ye *et al* [4]. They are starting with face image training set without labels and train a deep neural network iteratively model created was used to choose examples for further manual annotation. After following active learning strategy, Value of Information criterion is derived to actively select candidate annotation images. This model reaches the coverage of 70.7% with a precision of 95%.

MSR Image Recognition Challenge by J. Li *et al* introduces a knowledge base which has an idea to assign each face unique entity key and provide large dataset consisting of about 100,000 famous persons with around 100 images per person (MsCeleb) [5]. Method achieved coverage of 46.1% at 95% precision on the random dataset and 33% at 95% precision on the hard set of their challenge. Authors proposed a method consisting of two stages to learn robust human face representations for effective recognition of human faces. First stage in the training set is cleaning the noisy data because dataset is taken from the internet so images without faces can appear. In order to do so, a deep neural network was trained on existing dataset.

S. Chintalapati and M. V. Raghunadh used SVM and Bayesian classifier for automated attendance system based on face detection and recognition [6]. They proved that these classifiers are better when compared to other distance classifiers. This system automatically detects the student which enters the classroom and marks the attendance if recognizes him. One of the failures of the system is recognizing faces only up to 30 degrees angle variations.

3. Methods and Materials

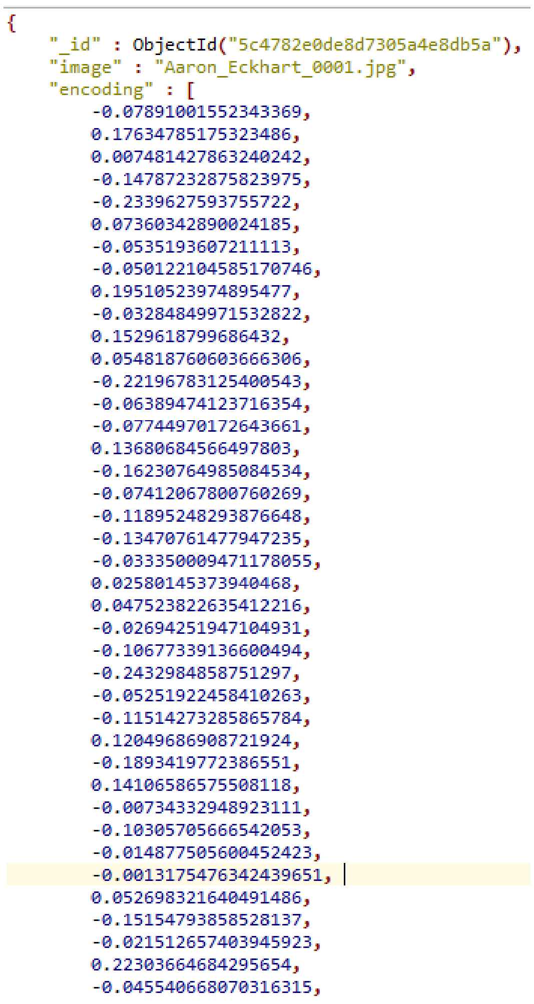
Dataset to be used is Labeled Faces in the Wild [7] which is a database of face photographs designed for studying the problem of unconstrained face recognition. It contains more than 13,000 pictures collected from the web. Each image has been labeled with the name of the person on it. There are 1680 different persons in images. Fig.1 shows samples from LFW dataset.

Fig.1. Samples from dataset

These images need to be processed in order to get numerical representation of faces which is called feature vector. Feature vector consists of various numbers in a specific order which can be: height or width of face, width of lips, nose height etc… Final output of processed image needs to be an array with features which is shown in Fig.2. All features are stored in the mongo database for speed improvement. Python script iterates in a folder which has dataset images and stores one by one in a database with image name and features. Face Recognition library with deep learning is going to be used for this project. Deep learning model has an accuracy of 99.38% on Labeled Faces in Wild dataset. Features of face recognition library are finding faces in pictures, finding facial features in pictures and identifying faces in pictures. Once installed face recognition gives us two command line programs:

- face_recognition - recognize face on image
- face_detection - detect face in image

Face recognition process consists of two stages. This includes taking and preparing training dataset and integration into existing system. For testing purposes, data was collected at the university. These are images of students which were taken in the first year. Images are preprocessed and inserted into the mongo database by using python script for inserting images. After insertion, facebook images of the same people were taken and tested by using python script for face recognition.



```
{
    "_id" : ObjectId("5c4782e0de8d7305a4e8db5a"),
    "image" : "Aaron_Eckhart_0001.jpg",
    "encoding" : [
        -0.0789100155234369,
        0.17634785175323486,
        0.007481427863240242,
        -0.14787232875823975,
        -0.2339627593755722,
        0.0736034289024185,
        -0.0535193607211113,
        -0.050122104585170746,
        0.1951052397489547,
        -0.03284849971532822,
        0.152961879968643,
        0.054818760603666306,
        -0.22196783125400543,
        -0.06389474123716354,
        -0.07744970172643661,
        0.1368068456649780,
        -0.16230764985084534,
        -0.07412067800760269,
        -0.11895248293876648,
        -0.13470761477947235,
        -0.033350009471178055,
        0.0258014537394046,
        0.04752382263541221,
        -0.02694251947104931,
        -0.10677339136600494,
        -0.2432984858751297,
        -0.05251922458410263,
        -0.11514273285865784,
        0.1204968690872192,
        -0.189341977238655,
        0.1410658657550811,
        -0.00734332948923111,
        -0.10305705666542053,
        -0.014877505600452423,
        -0.0013175476342439651,
        0.05269832164049148,
        -0.15154793858528137,
        -0.021512657403945923,
        0.2230366468429565,
        -0.045540668070316315,
```

Fig.2. Array of features

3.1 Data preprocessing

Implemented system is going to use monitoring cameras at the entrance. It means that we could have some kind of network or other problems and taken images could be blurry, so we have to include such images in training dataset, Fig.3. Persons entering the building can be photographed from different angles, so these kind of photos should be included also in training dataset, Fig 4.

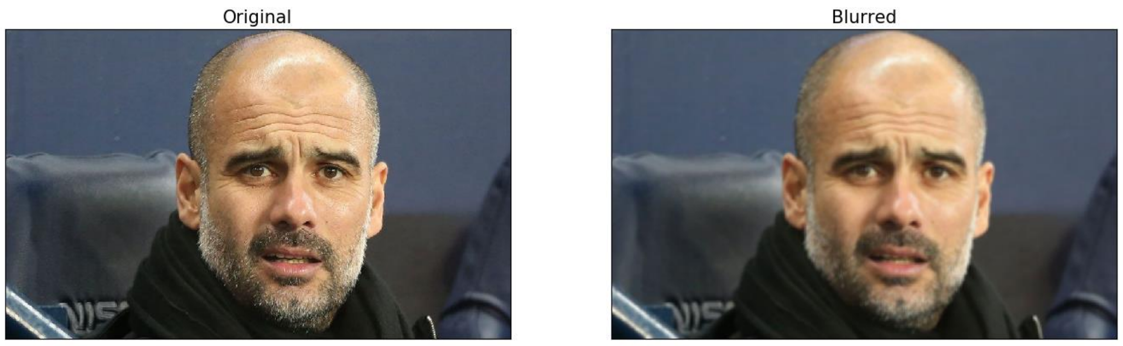


Fig.3. Original and Blurred image

Fig.3 shows an example of original and blurred images. If we have perfect conditions we would have a picture like the original one, but if the system experiences network issues we might have a blurry image like the one represented. System has to be ready to respond accordingly to these kinds of issues. Python script was written using OpenCV [8] interface to generate blurry images out of the original ones.

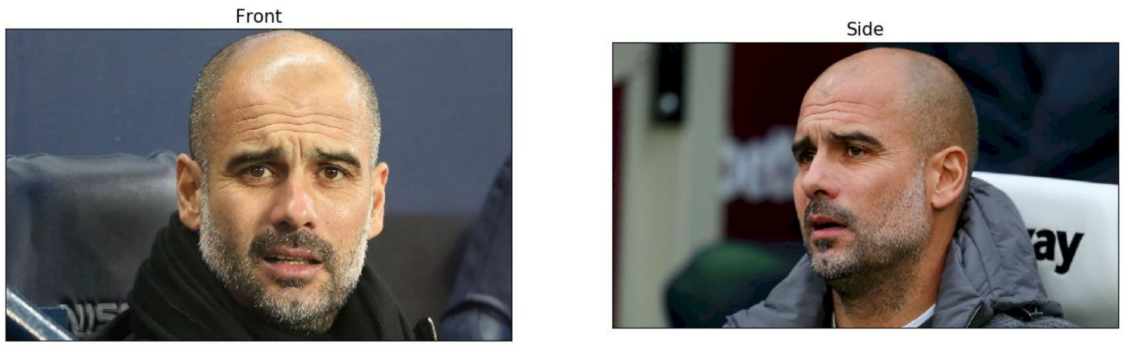


Fig.4. Image from front and side

In Fig.5 we can see facial features drawn on picture of Pep Guardiola. Most important features are shown: eyes, nose, mouth and chin location. These features are used when recognizing people on images. Face recognition library contains script face_landmarks for detecting facial landmarks and positioning the face based on them.

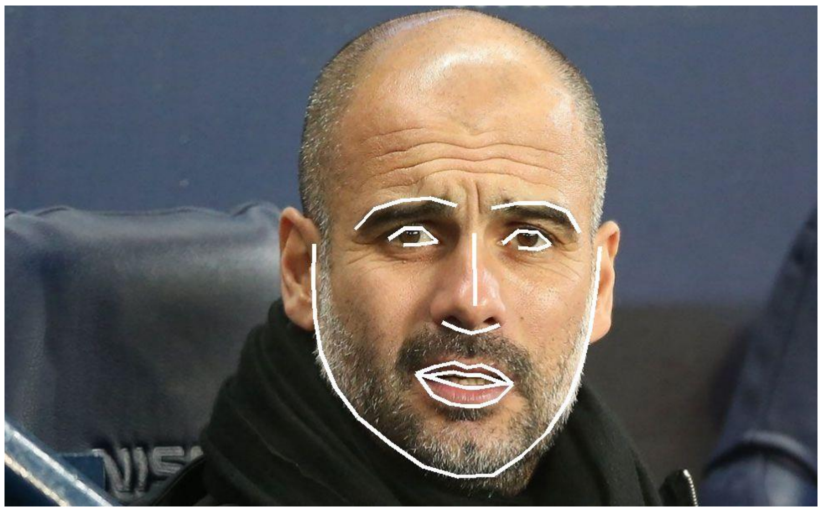


Fig.5. Facial features

3.2 Usage Workflow

Application usage workflow is represented on Fig.6. Images need to be collected into a single folder, so that insertion helper scripts can be run. For multiple insertions we need to pass a folder of images to script, which iterates through these images, creates encodings and inserts them into database. Single insertion script accepts an image as a parameter, encodes it and inserts into a database. Last step is recognizing images, the script accepts an image which needs to be recognized and iterates through mongoDB collection and looks for matching images.

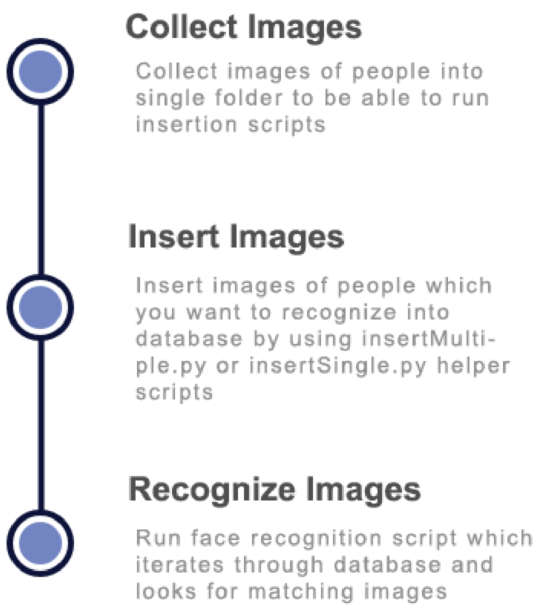

Collect Images
Collect images of people into single folder to be able to run insertion scripts
Insert Images
Insert images of people which you want to recognize into database by using insertMulti-ple.py or insertSingle.py helper scripts
Recognize Images
Run face recognition script which iterates through database and looks for matching images

Fig.6. Usage workflow

3.3 Face Recognition Library

Face recognition library which we are using is built using dlib’s face recognition with deep learning. We can install library by using a python package installer. Once installed we are provided with two command-line programs : *face_recognition* and *face_detection*. Face recognition recognizes faces in a photograph or in a folder of photos. There should be two folders, one containing known people and second which contains photos of people which we want to recognize. Face recognition program is run with two parameters which are the names of these folders. Face detection program finds pixel coordinates of faces. It takes a folder with images as parameter and at the end prints one line for each face that was detected. There is also an option to speed up the overall process by doing a recognition with multiple CPU cores. For example if we have 8 core CPU, we can process 8 times as many images in the same amount of time.

Dlib is a toolkit written in C++ and contains ML algorithms and tools for solving real world problems. The most important thing is that dlib is an open source library which enables anyone to use it anywhere, free of any charge. Some of the dlib’s features are Deep Learning, Multiclass SVM, Image Processing etc.. Our library uses Image Processing tool for face recognition built by using deep learning tools from dlib.

3.4 Helper Scripts

There are multiple helper scripts which enable user to insert multiple or single image into database and recognize faces. The most important parts of scripts are represented in the following lines.

insertMultiple.py

```
for image in images:
  current_image = face_recognition.load_image_file("images/" + image)
  encodings = face_recognition.face_encodings(current_image)
  if len(encodings) > 0:
    current_image_encoded = encodings[0]
    num_of_images+=1
  else:
   print("No faces found in the image " + image)
   num_of_not_found+=1
   continue
  mydict = { "image": image, "encoding": current_image_encoded.tolist() }
  x = mycol.insert_one(mydict)
  print(image + " inserted")
```

Code snippet above loads images from the folder, encodes them and inserts them into mongoDB. We have to provide the name of the folder which contains images and simply run the script.

faceRecognition.py

```
unknown_image = face_recognition.load_image_file("image.jpg")
unknown_face_encoding = face_recognition.face_encodings(unknown_image)[0]
known_faces = []
names = []
for x in mycol.find():
 known_faces.append(x['encoding'])
 names.append(x['image'])
results = face_recognition.compare_faces(known_faces, unknown_face_encoding)
for x in range(len(known_faces)):
 if results[x] == True:
   print("Recognized: " + names[x])
 else:
   print("Failed: " + names[x])
```

Code represented above does face recognition. It takes an image of the person which we want to recognize, encodes it, loops through face encodings from the database and checks if a person exists in the database.

4. Results

By using a custom dataset which was collected from our university. Students' images were taken and tested on created scripts. From these tests we have obtained accuracy of 90.9% when testing on images found on Facebook. There were some problems when recognizing people from different angles, but this can be material for further study. Images of people are not shown because they did not agree to publish their images.

Since face_recognition python library has a pre-trained model there is no need for additional training. Improvement is that all images are inserted in mongoDB with image name and face encodings array. Fig.5 shows one part of mongoDB record. Python script for inserting images in mongoDB is written and it takes a folder with images and inserts one by one in the database. In our testing environment LFW dataset is used and all images are collected into a single folder. Number of images inserted in the database is 4014. There are also images on which faces are not recognized. Unrecognized images number is 21. So, if we take into consideration that 4014 images are inserted and 21 are unrecognized which means that more than 99% images were recognized. Example of such an image is shown in Fig.6. Execution time of the script is about 35 minutes for LFW Dataset.

Final result of this research would be access control application. Application can be installed on Raspberry Pi which has a camera installed. All assets that are necessary for access control application to be fully functional can be installed on Raspberry. These are mongoDB, python and python libraries. Overall process is not so challenging, so that we do not need anything better than Raspberry Pi 3 B+ which is a model that we used while testing the application.

5. Conclusion

The aim of this study was to make the attendance checking process a lot easier for companies and schools by using biometrics. Every employee or student would be recorded by camera at the entrance and recorded in the system. For educational institutions cameras will be installed in classrooms so that the system can make multiple shots during lessons. This research was successful because it made the recognition process faster by using a document based database which is really fast.

This study will bring benefits for multiple groups. Benefit for schools is easier attendance recording and reducing waste of time at the beginning of the classes. Also students will not have a chance to avoid coming to classes because this system will not allow them to cheat. Similar benefit is for companies to track their employees coming and leaving time. Future research suggestions in this field are solving problems if a person is recorded from the side and possibly getting blurry images because of internet connection issues.

REFERENCES

[1] B. T. Liyew and P. Hazari, "A Survey on Face Recognition based Students Attendance System."

[2] X. Wu, R. He, Z. Sun, and T. Tan, "A Light CNN for Deep Face Representation With Noisy Labels," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 11. pp. 2884–2896, 2018, doi: 10.1109/tifs.2018.2833032.

[3] M. Arsenovic, S. Sladojevic, A. Anderla, and D. Stefanovic, "FaceTime — Deep learning based face recognition attendance system," *2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY)*. 2017, doi: 10.1109/sisy.2017.8080587.

[4] H. Ye *et al.*, "Face Recognition via Active Annotation and Learning," *Proceedings of the 2016 ACM on Multimedia Conference - MM '16*. 2016, doi: 10.1145/2964284.2984059.

[5] J. Li *et al.*, "Robust Face Recognition with Deep Multi-View Representation Learning," *Proceedings of the 2016 ACM on Multimedia Conference - MM '16*. 2016, doi: 10.1145/2964284.2984061.

[6] S. Chintalapati and M. V. Raghunadh, "Automated attendance management system based on face recognition algorithms," *2013 IEEE International Conference on Computational Intelligence and Computing Research*. 2013, doi: 10.1109/iccic.2013.6724266.

[7] "LFW Face Database : Main." [Online]. Available: http://vis-www.cs.umass.edu/lfw/. [Accessed: 19-Jan-2019].

[8] "OpenCV library." [Online]. Available: https://opencv.org/. [Accessed: 09-Jan-2019].