SAKARYA UNIVERSITY

Information Systems Engineering Department

GRADUATION PROJECT
ISE402

Deep Learning for Distributed Computer Vision Systems

Raport II/III

Supervised By:

Prof.Dr. İSMAİL HAKKI CEDİMOĞLU

cedim@sakarya.edu.tr

Written By:

ARSENE ADJEVI 0000-0003-4305-3914

arsene.adjevi@ogr.sakarya.edu.tr

Date: May 22, 2022

Contents

Abstract

Deep learning (DL) has seen great success in the computer vision (CV) field, and related techniques have been used in security, healthcare, remote sensing, and many other areas. As a parallel development, visual data has become universal in daily life, easily generated by ubiquitous low-cost cameras. Therefore, exploring DL-based CV may yield useful information about objects, such as their number, locations, distribution, motion, etc. Intuitively, DL-based CV can also facilitate and improve the designs of wireless communications, especially in dynamic network scenarios.. However, many multi-camera DL based computer vision algorithms assume that the information from all cameras is losslessly communicated. a central processor that solves the problem. This assumption is unrealistic for emerging wireless camera networks, which may contain processors with limited ca- pability, antennas with limited power, and batteries with limited life. In this project we will overview algorithms that solve DL based computer vision problems in a distributed manner, making them well-suited for implementation on a visual sensor network.

Keywords: *Computer Vision,Deep Learning,Networks programming,Distributed Systems.*

1 Introduction

Computer Vision, often abbreviated as CV, is defined as a field of study that seeks to develop techniques to help computers see and with the aid of multiple Deep Learning algorithms to understand the content of digital images such as photographs and videos and to predict what it is.This assumption is unrealistic for emerging wireless camera networks, which may contain processors with limited capability, antennas with limited power, and batteries with limited life.

2 Computer Vision and Deep learning

Computer vision is one of the capabilities which allows the machines to replicate this power. And using Deep Learning, we are enhancing our command and making advancements in this field.

2.1 Video Analytic Using Deep Learning

Videos are not new to us. We record videos using our phone, laptops, hand cameras, and so on. YouTube is one of the biggest sources of videos. Advertisements, movies, sports, social media uploads, TikTok videos, and so on are getting created every second. And by analyzing them, we can uncover a lot of insights about the behavior, interactions, timings, and sequence of things. A very powerful medium indeed! There can be multiple approaches to design a video analytics solution. We can consider video as a collection of frames and then perform analytics by treating the frames as individual images. Or we can also add an additional dimension of sound to it. In this book, we are concentrating our efforts on images alone, and sound is not included. We will now explore the various use cases of video analytics in the next section.

2.2 Use cases of video analytics

Videos are a rich source of knowledge and information. We can utilize Deep Learning–based capabilities across domains and business functions. Some of them are listed as follows[see Ver21, page 223]:

- ***Real-time face detection***
  Real-time face detection can be done using video analytics, allowing us to detect and recognize the faces. It has huge benefits and applications across multiple domains.
- ***In disaster management***
  video analytics can play a significant role. Consider this. In a flood-like situation, by analyzing videos of the actual area, the rescue team can identify the zones they should concentrate on. It will help reduce the time to action which directly leads to more lives saved.
- ***Similarly, for crowd management***
  video analytics plays an important role. We can identify the concentration of the population and the eminent dangers in that situation. The respective team can analyze the videos or a real-time streaming of video using cameras. And a suitable action can be taken to prevent any mishappening.
- **By *analyzing the social media videos***
  the marketing teams can improve the contents. The marketing teams can even analyze the contents of the competitors and tweak their business plan accordingly as per the business needs.
- ***For object detection and object tracking***
  video analytics can quickly come up with a decision if an object is present in the video or not.

This can save manual efforts. For example, if we have a collection of videos of different cars and we wish to classify them in different brands, the manual process will be to open each and every video and then take a decision – which is both time-consuming and error-prone. Using Deep Learning–based video classification, this entire process can be automated.

3 Face Recognition

Face recognition is nothing new. We are born with a natural capability to differentiate and recognize faces. It is a trivial task for us. We can recognize the people we know in any kind of background, different lights, hair color,with cap or sunglasses, and so on. Even if a person has aged or has a beard, we can recognize them. Amazing! Now we attempt to train the Deep Learning algorithms to achieve the same feat. A task so trivial and effortless for us is not an easy one for the machine. In Figure 6-1, we are having a face, then we are detecting a face, and then recognizing a face.

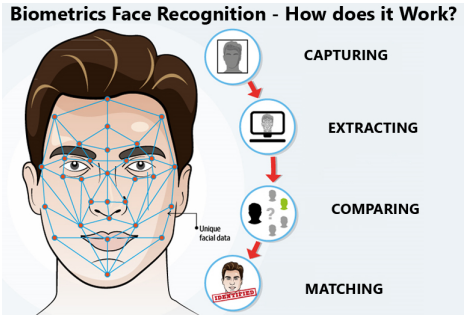

Figure 1: *We have a face initially and face recognition steps*

We can consider face recognition as a special case of object detection. Instead of discovering cars and cats, we are identifying people. But the problem is simpler; we have only the class of object to be detected – "face." But face detection is not the end state. We have to put a name to that face too, which is not trivial. Moreover, the face can be at any angle; a face can have different backgrounds. So, it is not an easy task. Also, we might be discovering faces in photographs or in videos. Deep Learning algorithms can help us in developing such capabilities. Deep Learning–based algorithms can leverage the computation power, advanced mathematical foundation, and the millions of data points or faces to train better models for face recognition.
Before we dive into the concepts and implementation of face recognition, we will explore the various use cases for this capability.

3.1 Application of face recognition

Face recognition is a pretty exciting technology having applications across domains and processes. Some of the key uses are[see Ver21, page 190]:

- **Security management:** The face recognition solutions are applicable for both online and offline security systems. Security services, police departments, and secret services utilize the power of machine learning–based face recognition techniques to trace the antisocial elements. Passport verification can happen quicker and in a much more robust fashion. Many countries do maintain a database of criminals' photographs which acts as a starting point to trace the culprits. The technology saves really a great deal of time and energy and allows the investigators to focus their energies on other areas.
- **Identity verification** is another big area employing face recognition techniques. One of the most famous examples of ID verification is smartphones. Face ID is used in iPhones and the phone unlocks. Face recognition is being used by online channels and social media to check the identity of the person trying to access the account.
- **It is used by retailers** to know when individuals with not-so-good history have entered the premises.When shoplifters, criminals, or fraudsters enter the stores, they act as a threat. Retailers can identify them and take immediate actions to prevent any crime.
- **Marketing** becomes much sharper if the business knows the age, gender, and facial expressions of the customer. Giant screens can be installed (in fact have been done) to identify the target audience.
- **Consumer experience** is improved when the consumer-product interaction is analyzed. Expressions of people when they touch the products or try them capture the real-world interactions. The data acts as a gold mine for the product teams to make necessary amendments to the product features. At the same time, the operations and in-store team can make the overall shopping experience more enjoyable and interesting.
- **Access to offices, airports, buildings, warehouses, and parking lots** can be automated with no human intervention. A security camera takes a picture and compares it with the database to ensure authenticity.

3.2 Process of face recognition

We click pictures from our phones and cameras. The photos are taken of various occasions – marriage, graduation, trips, holidays, conferences, and so on. When we upload the pictures on social media, it automatically detects the face and recognizes who the person is. An algorithm works in the background and does the magic. The algorithm is not only able to detect the face but will put a name to it from all the other faces in the background. We are studying the similar process in this section.Broadly, we can have these four steps around face recognition as shown in *Figure III.2*

Figure 2: *Process followed in face detection – right from detection to recognition. We detect a face, perform alignment, extract the features, and finally recognize the face*

1. Face detection simply means to **locate** if there is a face or multiple faces in a photograph. And we will create a bounding box around it.
2. Once we have detected the face, we **normalize** the attributes of the face like the size and geometry. It is done so that it matches the facial database we have. We also reduce the effect of illumination, head movement, and so on.
3. Next, we **extract** the features from the face. Some of the distinguishing features are eyes, eyebrows, the nostrils, corners of the mouth, and so on.
4. And then we perform the face **recognition**. It means we match the face with the existing ones in the database. We might perform one of the two:
    - Verify the given face with a known identity. In simple terms, we want to know "Is this Mr. X?". It is a case of one-to-one relationship.
    - Or we might want to know "Who is this guy?," and in such a case we will have a one-to-many relationship.

Deep Learning is making its presence felt for face recognition too. Recall that face recognition is similar to any other image classification solution. But faces and attributes of features make face recognition and detection quite a special one.

We can use the standard Convolutional Neural Network for face recognition too. The layers of the network will behave and process the data similar to any other image analysis problem.Bu there are a plethora of solutions available, but the most famous are ***DeepFace, VGGFace, DeepID, and FaceNet*** as Deep Learning algorithms.

3.3 DeepFace solution by Facebook

DeepFace was proposed by researchers of Facebook AI Research (FAIR) in 2014.The actual paper can be accessed at `http://www.cs.toronto.edu/~ranzato/publications/taigman_cvpr14.pdf`[Tai+14]

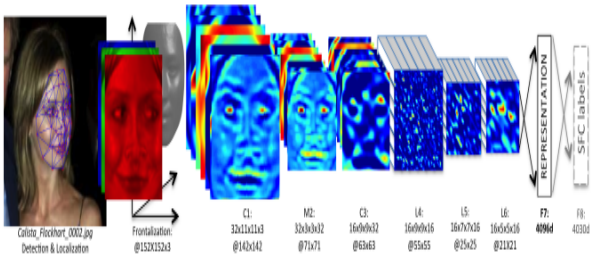

Figure 2. **Outline of the *DeepFace* architecture**. A front-end of a single convolution-pooling-convolution filtering on the rectified input, followed by three locally-connected layers and two fully-connected layers. Colors illustrate feature maps produced at each layer. The net includes more than 120 million parameters, where more than 95% come from the local and fully connected layers.

Figure 3: *DeepFace architecture is shown here. The figure has been taken from the original paper.*

In the architecture shown earlier, we can analyze the various layers and the processes of the network. DeepFace expects an input image as a 3D-aligned RGB image of 152x152. We will now explore this concept of 3D alignment in detail.The objective of alignment is to generate a front face from the input image. The complete process is shown in Figure 6-4 which is taken from the same paper.The objective of alignment is to generate a front face from the input image. The complete process is shown in Figure 6-4 which is taken from the same paper.

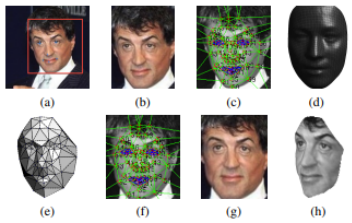

Figure 4: *Face alignment process used in DeepFace. The image has been taken from the original paper.*

- In the first step, we detect a face using six fiducial points. These *six fiducial* points are two eyes, tip of the nose, and three points on the lips. In *Figure 6-4*, it is depicted in step (a). This ***step detects the face in the image***.
- In the second step, as shown in step (b), we ***crop and generate the 2D face*** from the original image. We should note how the face has been cropped from the original image in this step.
- In the next steps, triangles are added on the contours to avoid discontinuities. We ***apply 67 fiducial points*** on the 2D-aligned crop with their corresponding Delaunay Triangulation. A 3D model is generated using a 2D to 3D generator, and the 67 points are plotted. It also allows us to align the out-of-plane rotation.
- Step (e) shows the visibility with respect to the ***fitted 2D-3D camera***,
- In step (f) we can observe ***the 67 fiducial points induced by the 3D model which are used to direct the piecewise affine wrapping***.

We will discuss the **Delaunay Triangulation** briefly now. For a given set "P" of discrete points in a plane, in triangulation DT no point is inside the circumcircle of any triangle in Delaunay Triangulation. Hence, it maximizes the minimum angles of all the triangles in the triangulation. We are showing the phenomenon in *Figure 5*

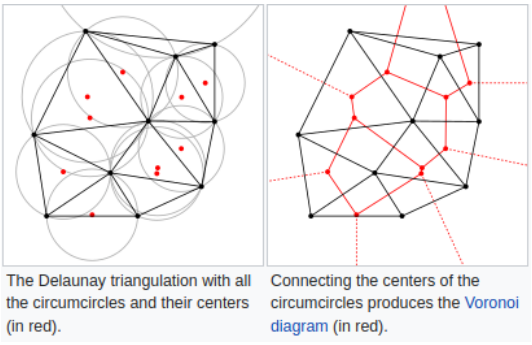

Figure 5: *Delaunay Triangulation. Image source:https://en.wikipedia.org/wiki/Delaunay_triangulation*

Finally, we do a final ***frontalized crop***. It is the final step which achieves the objective of ***3D frontalization***. Once the step of 3D frontalization is done, then the image is ready to be fed to the next steps in the network. An image of 152x152 size is the input image which is fed to the next layer. The next layer is the convolutional layer (C1) with 32 filters of size 11x11x3 followed by a 3x3 max pooling layer with a stride of 2 . Then the next layer is another convolution with 16 filters and size 9x9x16 like described in original paper [see Tai+14, Figure 2].Then we have three locally connected layers. We will discuss locally connected layers briefly as they are a bit different from fully connected layers.

Locally connected layers behave differently from ***fully connected layers***. For a fully connected layer, each neuron of the first layer is connected to the next layer. For locally connected layers, we have different types of filter in a different feature map. For example, when we are classifying if the image is of a face, we can search for the mouth only at the bottom of the image. So locally connected layers are handy if we know that a feature should be restricted within a small space and there is no need to search for that feature across the entire image.

In DeepFace, we have ***locally connected layers***, and hence we can improve the model as we can differentiate between facial regions based on different types of feature maps. The penultimate layer is a fully connected layer which is used for face representation. ***The final layer is a softmax fully connected layer*** to do the classification. The total number of parameters is 120 million. *Dropout* is being used as a regularization technique but is done only for the final fully connected layers. We also ***normalize the features between 0 and 1 and do an L2 normalization***. The network generates quite sparse feature maps during the training, primarily since ***ReLU has been used as the activation function***. The validation was done on the ***LFW (Labeled Faces in the Wild) dataset and SFC dataset***. LFW contains more than 13,000 web images of more than 5700 celebrities. SFC is the dataset by Facebook itself having 4.4 million images of 4030 people each having 800 to 1200 facial images. The ROC curves for both the datasets are below:

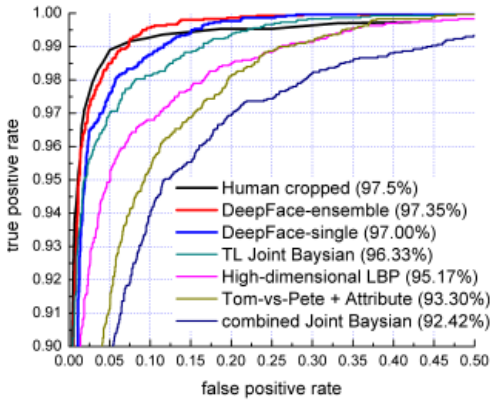

Figure 6: *The ROC curves on the LFW dataset. Best viewed in color taken from the original paper[see Tai+14, Figure 3]*

DeepFace is one of the novel face recognition

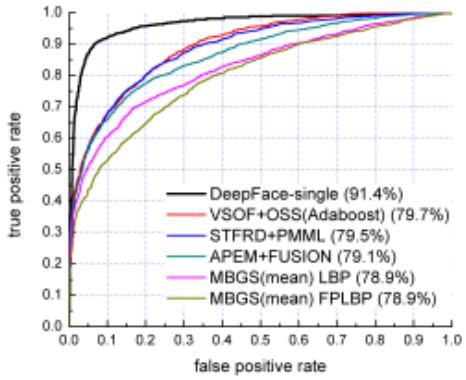

Figure 7: *The ROC curves on the YTF dataset. Best viewed in color taken from the original paper[see Tai+14, Figure 4]*

modes. It has ***more than 99.5% accuracy on the LFW dataset***. It is able to resolve issues with pose, expression, or light intensity in the background. 3D alignment is quite a unique methodology which further enhances accuracy. The architecture has performed really well on LFW and YouTube Faces dataset (YTF). We have finished discussing the DeepFace architecture now.

3.4 FaceNet for face recognition

FaceNet was proposed by Google researchers Florian Schroff, Dmitry Kalenichenko, and James Philbin in 2015. The original paper is "FaceNet: A Unified Embedding for Face Recognition and Clustering" and can be accessed at `https://arxiv.org/pdf/1503.03832.pdf`[SKP15]
FaceNet does not recommend a completely new set of algorithms or complex mathematical calculations to perform the face recognition tasks. The concept is rather simple.

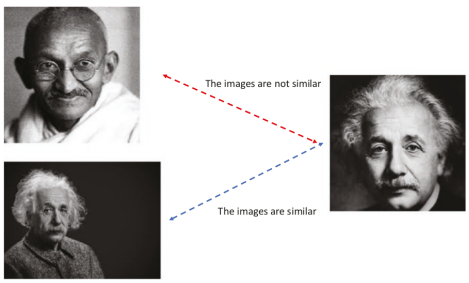

Figure 8: *The images of Einstein will be similar to each other, and hence the distance between them will be less, while the image of Gandhi will be at a distance[see Ver21, Page 200]*

All the images of faces are first represented in a ***Euclidean space***. And then we ***calculate the similarity between faces by calculating the respective distances***. Consider this, if we have an image, Image 1 of Mr. X, then all the images or faces of Mr. X will be closer to Image 1 rather than Image 2 of Mr. Y. The concept is shown on *Figure 8*.

The preceding concept is simpler to understand. We will understand the architecture in detail now. As shown in Figure below, we can examine the complete architecture. The image has been taken from the original paper itself. FaceNet

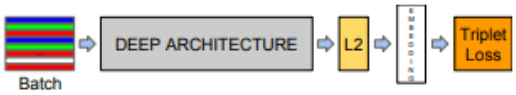

Figure 9: *FaceNet Model structure. The network consists of a batch input layer and a deep CNN followed by L2 normalization, which results in the face embedding. This is followed by the triplet loss during training. The image has been taken from the original paper [see SKP15, Figure 2]*

implements ***1x1 convolutions to decrease the number of parameters***. The output of these Deep Learning models is an ***embedding of images***. L2 normalization is performed on the output. These embeddings are quite a useful addition. FaceNet understands the respective mappings from the facial images and then creates embeddings.
Once the embeddings are successfully done, we can simply go ahead, and with the help of newly created embeddings as the feature vector, we can use any standard machine learning technique. ***The usage of embeddings is the prime difference between FaceNet and other methodologies as other solutions generally implement customized layers for facial verifications***.
The created embeddings are then fed to calculate the loss. As discussed earlier, the images are represented in a Euclidean space.***The loss function aims to make the squared distance between two image embeddings of similar images small, whereas the squared distance between different images is large***. In other words, the squared distance between the respective embeddings will decide the similarity between the faces.
Triplet loss is shown in Figure below; the image has been taken from the original paper itself.

Figure 10: *The Triplet Loss minimizes the distance between an anchor and a positive, both of which have the same identity, and maximizes the distance between the anchor and a negative of a different identity.The image has been taken from the original paper [see SKP15, Figure 3]*

The intuition is that we want the images of the same person Mr. X closer to each other. Let us call that Image 1 as the anchor image. All the other images of Mr. X are called the positive images. The images of Mr. Y are referred to as negative images. Hence, as per triplet loss, we want the distance between the embeddings of the anchor image and positive image to be less as compared to the distance between embeddings of the anchor image and negative image.We would want to achieve Equation:

$$\left\|x_i^a - x_i^p\right\|_2^2 + \alpha < \left\|x_i^a - x_i^n\right\|_2^2, \forall (x_i^a, x_i^p, x_i^n) \in \tau$$

where Anchor image is x_i^a .
Positive image is x_i^p .
Negative image is x_i^n , so basically x_i is an image.
α is a margin that is enforced between positive and negative pairs. It is the threshold we set, and it signifies the difference between the respective pairs of images.
τ is the set of all the possible triplets in the training set and has cardinality n.
Mathematically, triplet loss can be represented as in below. It is the loss which we wish to minimize.

$$\sum_i^N [\left\|f(x_i^a) - f(x_i^p)\right\|_2^2 - \left\|f(x_i^a) - f(x_i^n)\right\|_2^2 + \alpha]$$

In the preceding equations, the embedding of an image is represented by $f(x)$ such that $x \in R$. It embeds an image x into a d-dimensional Euclidean space.$f(x_i)$ is the embedding of an image which is in the form of a vector of size 128.
The solution is dependent on the selection of the image pairs. There can be image pairs which the network will be able to pass. In other words, they will satisfy the condition of the loss. These image pairs might not add much to the learning and might also result in slow convergence.
Mathematically, for an anchor image x_i^a , we would want to select a positive image x_i^p such that the similarity is maximum and select a negative image x in such that the similarity is minimum. In other words, we wish to have $argmax\ x_i^p \left\|f(x_i^a) - f(x_i^p)\right\|_2^2$ which means given an anchor image x_i^a we wish to have a positive image x_i^p such that the distance is maximum. Similarly, for a given anchor image x_i^a , we wish to get a negative image x in such that the distance is minimum which is represented as $argmin\ x_i^n \left\|f(x_i^a) - f(x_i^n)\right\|_2^2$ Now, making this choice is not an easy task. During training, it is ensured that positive and negative are chosen as per the maximum and minimum functions given earlier on the mini-batch.***SGD (Stochastic Gradient Descent) with Adagrad(Adaptive Gradients)*** is used for training.

FaceNet is a novel solution as it directly learns an embedding into the Euclidean space for face verification. The model is robust enough to be not affected by the pose, lighting, occlusion, or age of the faces.

3.5 VGGFace and VGGFace2 Models[Bro19]

The VGGFace refers to a series of models developed for face recognition and demonstrated on benchmark computer vision datasets by members of the ***Visual Geometry Group (VGG)*** at the University of Oxford and it is first CNN Model specially a **Resnet-50** we have used for our code example to recognize faces on remote devices cameras over ***UDP (User Datagram protocol)***.
The **VGGFace model**, was described by Omkar Parkhi in the 2015 paper titled *Deep Face Recognition*[PVZ15]. A contribution of the paper was a description of how to develop a very large training dataset, required to train modern-convolutional-neural-network-based face recognition systems, to compete with the large datasets used to train models at Facebook and Google. A deep convolutional neural network architecture is used in the VGG style, with blocks of convolutional layers with small kernels and *ReLU activations* followed by max pooling layers, and the use of fully connected layers in the classifier end of the network.
Qiong Cao, et al. from the VGG describe a follow-up work in their 2017 paper titled ***VGGFace2: A Dataset For Recognizing Faces Across Pose And Age.***[Cao+18] ***They describe VGGFace2 as a much larger dataset that they have collected for the intent of training and evaluating more effective face recognition models***.The paper focuses on how this dataset was collected, curated, and how images were prepared prior to modeling. Nevertheless, VGGFace2 has become the name to refer to the pre-trained mod-

els that have provided for face recognition, trained on this dataset. ***Models are trained on the dataset, specifically a ResNet-50 and a SqueezeNet-ResNet-50 model (called SE-ResNet-50 or SENet)***, and it is variations of these models that have been made available by the authors, along with the associated code. The models are evaluated on standard face recognition datasets, demonstrating then state-of-the-art performance.
A face embedding is predicted by a given model as a 2,048 length vector. The length of the vector is then normalized, e.g. to a length of 1 or unit norm using the L^2 vector norm (Euclidean distance from the origin). This is referred to as the face descriptor. The distance between face descriptors (or groups of face descriptors called a subject template) is calculated using the Cosine similarity.

4 Installing the Toolkits

We can't reasonably give an exhaustive guide for installing the toolkits on all systems and hardware. Instead, we'll provide step-by-step instructions for the specific operating system we'll use as the reference system. These steps, along with the minimum version numbers of the libraries, should be enough for most readers to get a working system in place.
To install the remaining packages, we need to go into a shell and execute the sequence of steps below in the order given:

Command Line

```
$ sudo apt - get update
$ sudo apt - get install python3 - pip
$ sudo apt - get install build-
essential python3 - dev
$ sudo apt - get install python3-
setuptools python3 - numpy
$ sudo apt - get install python3-
scipy libatlas - base - dev
$ sudo apt - get install python3-
matplotlib
$ pip3 install scikit-learn
$ pip3 install tensorflow
$ pip3 install pandas
$ pip3 install pillow
$ pip3 install h5py
$ pip3 install keras
$ pip3 install seaborn
$ pip3 install pickle
$ pip3 install keras_vggface
$ pip3 install opencv-python
```

5 Experiments results

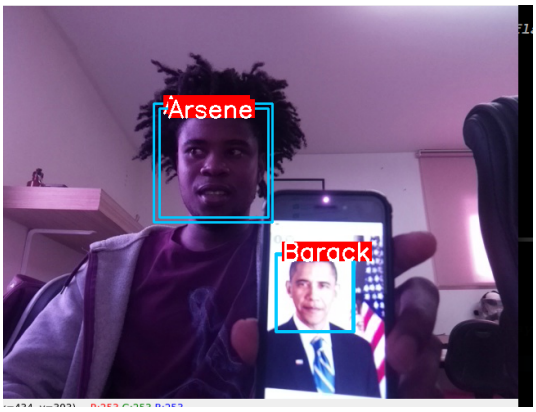

Figure 11: *My face was detected and recognized by the system.*

6 Conclusions

I have only included 8 people in this to fit the VGGFace Restnet network on this project . As you can imagining, as the number of people grows, the model will likely to confuse with two similar faces.We have also some issues with the jpeg format data.The jpeg buffer frame likely overflowing after some time .Other problem is that pickle library return some error after some time when try to deserialize the buffer frame.This mean this application at the time of this writing is not reliable at all and i keep work on it.You can find the source code at `https://github.com/cybdry/DeepFlicc`

References

[Tai+14] Yaniv Taigman et al. "DeepFace: Closing the Gap to Human-Level Performance in Face Verification". In: *Facebook AI Research* (2014).

[PVZ15] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. "Deep Face Recognition". In: *Proceedings of the British Machine Vision Conference (BMVC)*. Ed. by Mark W. Jones Xianghua Xie and Gary K. L. Tam. BMVA Press, Sept. 2015, pp. 41.1–41.12. ISBN: 1-901725-53-7. DOI: 10.5244/C.29.41. URL: https://dx.doi.org/10.5244/C.29.41.

[SKP15] Florian Schroff, Dmitry Kalenichenko, and James Philbin. “FaceNet: A Unified Embedding for Face Recognition and Clustering”. In: *arXiv:1503.03832v3* (2015).

[Cao+18] Qiong Cao et al. “VGGFace2: A dataset for recognising faces across pose and age”. In: May 2018. URL: https://arxiv.org/abs/1710.08092.

[Bro19] Jason Brownlee. *Image Classification, Object Detection and Face Recognition in Python*. 2019.

[Ver21] Vaihav Verhan. *Computer Vision Using Deep Learning:Neural Network Architectures with Python and Keras*. Apress, 2021.