זיהוי רגשות בזמן אמת

Real-Time

Emotion Detection

Supervisor: רן ברויאר | Ran Breuer

By: ענת וייברמן וטל אמיר | Anat Veikherman & Tal Amir

סימסטר רישום: חורף תשפ"ב

תאריך הגשה: ספטמבר 2022

Contents

Abstract

The goal of this project is to build a real-time emotion detection application. This application will record the participants in real-time, detect their faces and recognize their emotion. The idea can help in several domains, for example sales companies who want to test their new advertisement can present the advertisement to a group of potential clients and watch the emotions on their faces while they watch the advertisement. That way they could ask themselves if the emotions of the clients are what they expected for them to feel. With the information that they got from the application, they could improve the advertisement.

The challenge of the project is to detect faces and emotions of multiple people at once. Another challenge is to detect al of the participance in real-time and with no delays.

The face detection task

Face detection is a computer technology that identifies human faces in digital images.

It is a problem of object recognition that requires that both the location of each face in a photograph is identified (e.g. the position) and the extent of the face is localized (e.g. with a bounding box). Object recognition itself is a challenging problem, although in this case, it is similar as there is only one type of object, e.g. faces, to be localized, although faces can vary wildly.

Detecting faces in pictures can be complicated due to the variability of factors such as pose, expression, position and orientation, skin color and pixel values, the presence of glasses or facial hair, and differences in camera gain, lighting conditions and image resolution.

face detection methods can be broadly divided into two main groups:

- Feature-Based.
- Image-Based.

The feature-based face detection uses hand-crafted filters that search for and locate faces in photographs based on a deep knowledge of the domain. They can be very fast and very effective when the filters match, although they can fail dramatically when they don’t.

Alternately, image-based face detection is holistic and learns how to automatically locate and extract faces from the entire image. Neural networks fit into this class of methods.

Emotion Recognition Task

The Emotion Recognition Task is a task of measuring the recognition of seven basic facial emotional expressions (in addition to natural position):

- Anger - Lowered and burrowed eyebrows, intense gaze, raised chin.
- Happy - Raised corners of mouth into a smile.
- Surprise - Dropped jaw, raised brows, wide eyes.
- Fear - Open mouth, wide eyes, furrowed brows.
- Sadness - Furrowed brows, lip corner depressor.
- Disgust – Biting of the lips.

The task is to study the expressions depending on many factors to conclude what emotion the person is showing. Factors such as:

- Location of the eyebrows and eyes
- Position of the mouth
- Distinct changes of the facial features

Example of 7 expressions taken from Fer2013

Goals

- Experience new python frameworks for Image processing and learn about face detection and emotion recognition approaches today.
- Learn how to create client-server application using Django framework.
- The application should detect each person in the frame and save frames that have interesting emotions.
- Improve exiting emotion detection algorithm by adding multi person detection and improving time results in order to use it as a real – time application.

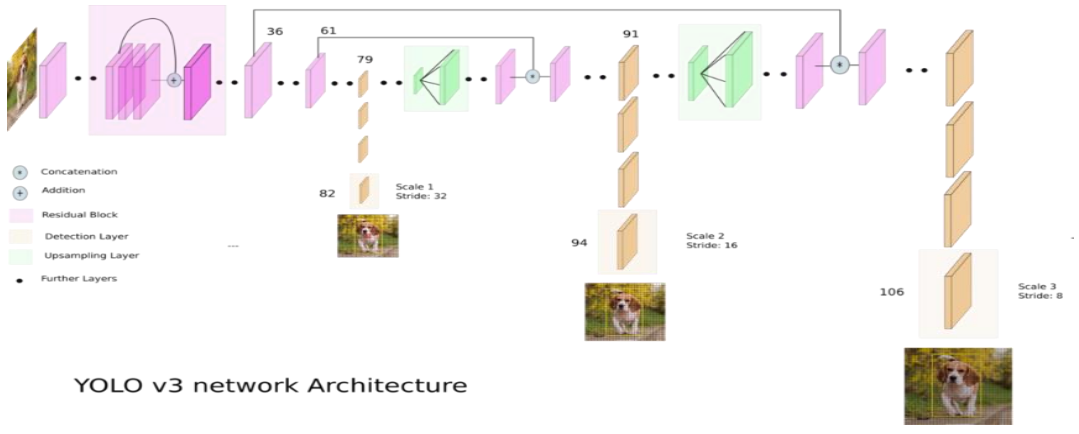
Yolo

Yolo method

YOLO [1]is an algorithm that uses neural networks to provide real-time object detection. This algorithm is popular because of its speed and accuracy. You Only Look Once - The algorithm requires only a single forward propagation through a neural network to detect objects. YOLO uses anchor boxes for object detection and classification, and detect objects that is suitable for the shape of the defined boxes.

YoloFace

YoloFace is based on YOLOv3 CNN. The YOLOv3 is a state-of-the-art, real-time object detection algorithm. As mentioned before, YOLO uses anchors for detection. Originally, YOLO performs well when facing normal size objects, but is incapable of detecting small objects. The accuracy decreases notably when dealing with objects that have large-scale changing like faces. YoloFace tries to improve the performance for face detection. The main change involves using anchor boxes more appropriate for face detection and a more precise regression loss function. The improved detector significantly increased accuracy while remaining fast detection speed.



YOLO v3 network Architecture

[1] https://arxiv.org/pdf/1506.02640.pdf

DeepFace

Deepface [2]is a lightweight face recognition and facial attribute analysis framework for python.

The framework has four main steps:

- Detect
- Align
- Represent
- Classify

Detect:

In this step, the algorithm first detects the face which it will than categorize.

The detection is made by first identifing the face in the input using six fiducial points. These six fiducial points are 2 eyes, tip of the nose and 3 points on the lips.

The DeepFace algorithm is only able to detect one face per image, which is a major limitation when the input contains more than one face.

Alignment:
The goal of the alignment part is to generate frontal face from the input image that may contain faces from different pose and angles. The method proposed in DeepFace paper used 3D frontalization of faces based on the fiducial (face feature points) to extract the frontal face.

Final aligned image

Image before alignment

Representation:

Human faces have to be represented before recognition can take place. The role played by representation is most important, and it probably exceeds that played by recognition, known also as classification or identification. The representations, learned through visual experience, are directly related to the context they operate within and the tasks they have to accomplish. The non-accidental properties of human faces are the particular dimensions

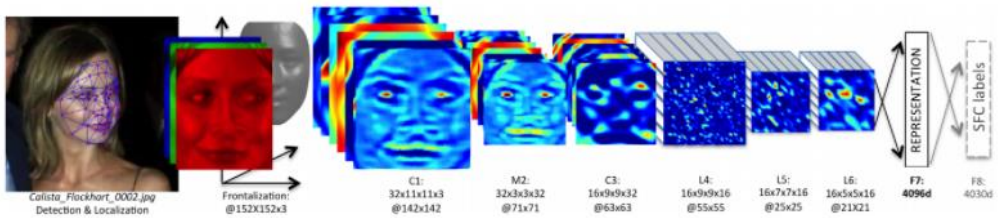
[2]Deepface article: https://www.cs.toronto.edu/~ranzato/publications/taigman_cvpr14.pdf

the human face representations become tuned for. Such properties eventually induce the features extracted to represent human faces. The features correspond to coordinate axes and they define the face space.

Classify:

The classify step is the last step of the DeepFace algorithm. In this step, the algorithm determines the emotion of the face in the input.

In the architecture the output of the last layer is the softmax layer K classes for the classification of the face.



DeepFace full architecture

Client Server

General Explanation

In general, the client is a computer application that request services from server processes. The server is a computer that provides the services to the client. Client-server is essentially a relationship between processes running on separate computers.

The client-server model is a distributed application framework dividing tasks between servers and clients, which either reside in the same system or communicate through a computer network or the Internet. The client relies on sending a request to another program in order to access a service made available by a server. The server runs one or more programs that share resources with and distribute work among clients.

The client server relationship communicates in a request–response messaging pattern and must adhere to a common communications protocol, which formally defines the rules, language, and dialog patterns to be used.

Django

Django [3]is a high-level Python web framework that enables rapid development of secure and maintainable websites. Django handles much of the hassle of web development, so the developer can focus on writing the app instead of reinventing the wheel.

Why do we need a server?

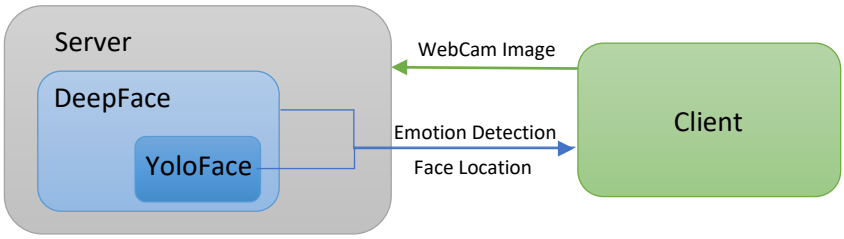
- Easy to maintain the algorithm in the future. If we were to update the algorithm or improve it in the future, we wouldn't need to bother the client with a new version, we would only have to update the algorithm on the server.
- Multi-client access to same algorithm. Since the algorithm does not run on the client's local computer, multiple clients can send requests to the server.
- Using a remote server, we can upgrade our hardware and computational resources.
- The application can be used from any computer or mobile device.

[3] https://www.djangoproject.com/

Our Client-Server implementation

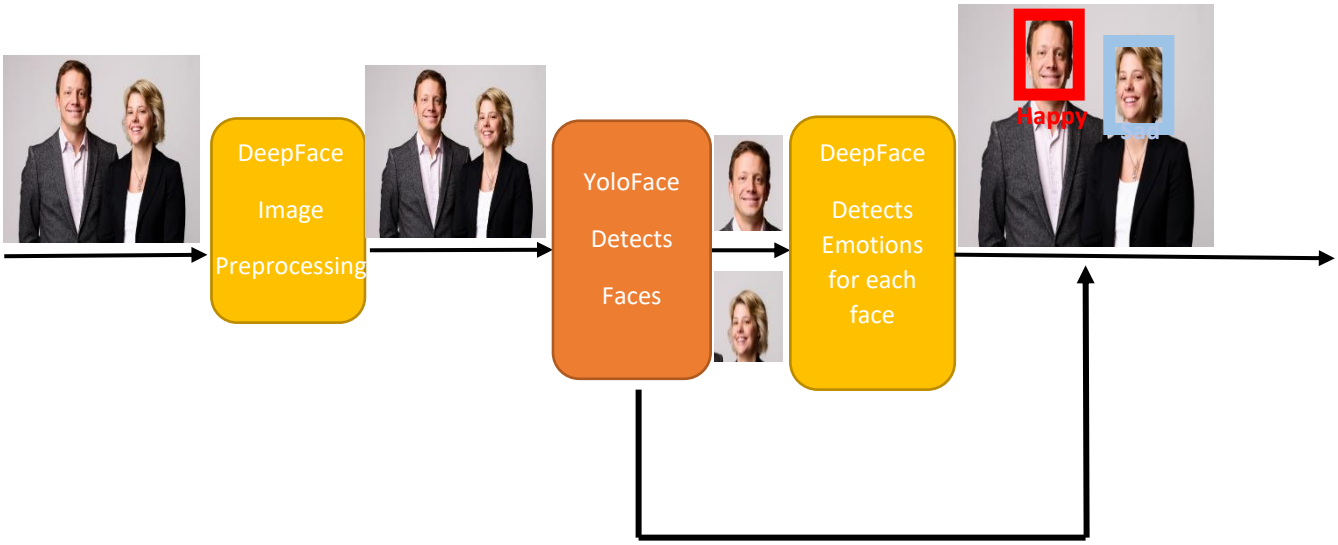
In our case, we use a client-server architecture to communicate between the web camera, which is the client, and the algorithm stored on the server.

This is a basic schema of our use of the server:



As can be seen in the schema, the camera is located on the client side of the application, photographing the person. The client then sends the image to the server, which applies the algorithm to the image. The server then sends the result back to the client, namely the facial location and emotion.

Our Algorithm



First, we receive a picture taken by user's web cam (front side). Afterwards, this picture is sent to our server for analyzing. The server is in charge of the decisions about the emotions of the people being captured. The original photo, is firstly proceeded to the DeepFace algorithm and being pre processed to be fitted inside the neural network. Pre processing includes resizing, cropping, and changing colors.

Then, before classifying the emotions, we need to detect faces in the image. Here YoloFace comes in charge and search for human faces in the image. YoloFace returns the coordinates of the faces and we crop each one of them and organize them in a list.

Each of the faces is fed into the final DeepFace stage and each face receives probability for each one of the possible emotions. We choose the emotion that gets the highest one and return it with our decision to the client side.

Our improvements

Classify multiple people emotions:
In this project we adjusted the model to work with multi persons. As we described above, the DeepFace framework is unable to classify more than one person in an image, making it very difficult and time consuming when working with images that contain multiple people. The algorithm would have taken too much time to apply if it had been performed on each individual at a time, so it was crucial for the real-time performance of the algorithm to run it on multiple people at one time.

Merging image processing:
We improved the time analysis of the image by merging the YoloFace framework with DeepFace framework. By doing this merge, our algorithm, instead of analyzing the image twice (with YoloFace and then with DeepFace), it analyzes the image only once. The time it takes to classify the image was significantly reduced due to this change.

Time improvement by Image scaling
Another improve of the time was reducing the time it takes to send a message from the client to the server. There is an overhead of sending the image from the client to until it arrives the server. As the image is smaller, this time is reduced. Thus, we resized the image from 460x640 to 360x480 so that the results and classification would not be affected.

Real-time Processing
Our last improvement was the real-time Performance of the algorithm. For this improvement we consider two options. One way was to go over each face in an image in Round Robin and detect a different face each time. This relies on the assumption that people are not moving so much when they are looking on a camera, so that we don't need to classify the same face multiple times. The second option was to send the frame from the camera every 20 frames instead of every frame. This is also relying on the same assumption that people are not changing their faces and emotions frequently when looking at a camera. Finally, we chose the second option because we wanted the ability to classify multiple people at once.

Results

After we developed the algorithm, we wanted to check its behaviors on real image.

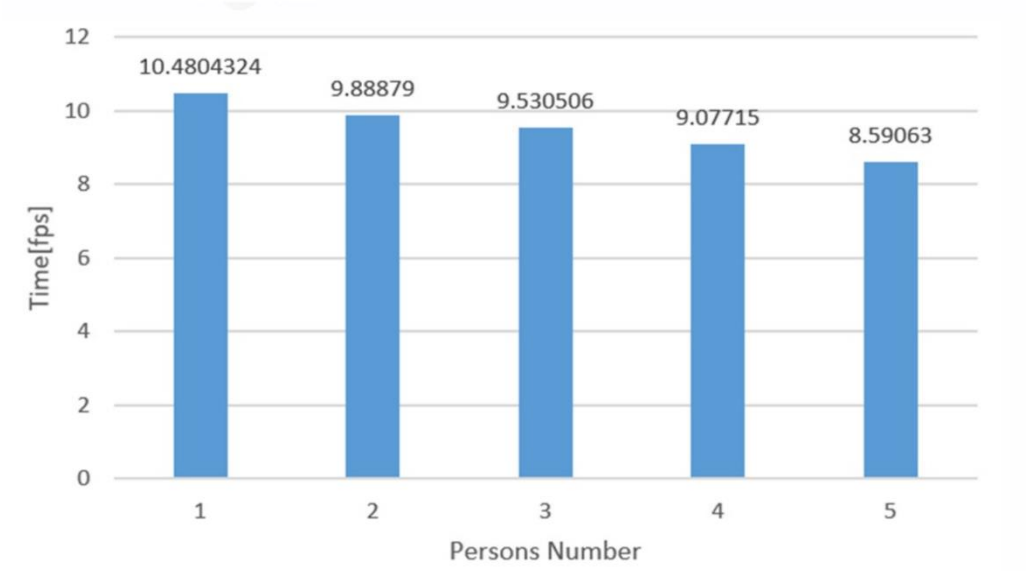
The most important thing that we focused on was fps – frame per second.

The reason is, that our application should work in “real time” and the improvements we made supposed to reach this goal.

The results below is for the algorithm running with the following computational resources - Intel(R) Core(TM) i5-8250U CPU@ 1.60GHz 1.80 GHz.

fps for different person amount

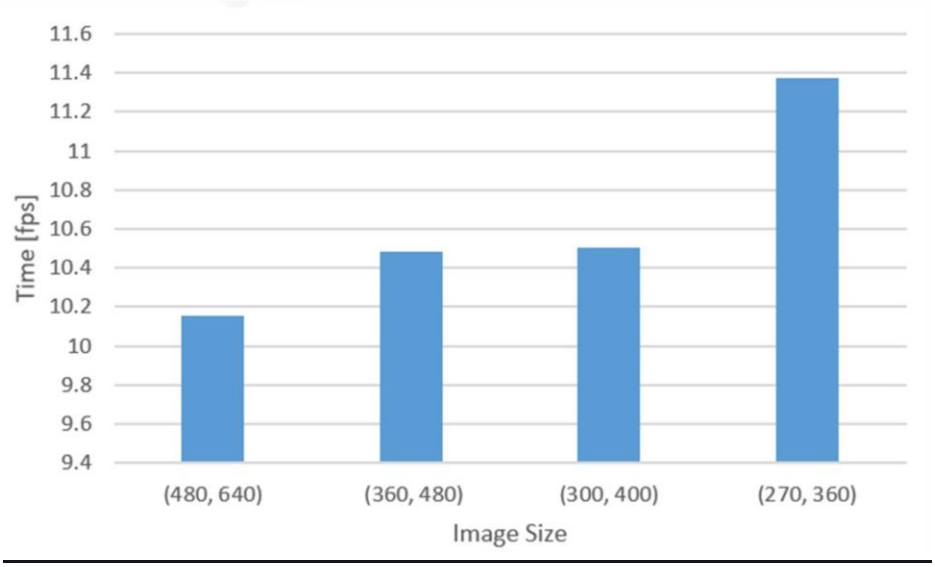
Here we take images size of 360,480 and check how fps changes for different persons number in the frame:



As we can see, as long as the number of persons in the frame gets higher, fps gets lower as we expected. But, it is important to mention that the fps reduces but not dramatically. Furthermore, if we calculate the time-per-person, we can see that in decreases as there are more faces in the frame. From here, we can conclude that most of the time is wasted on operations that does not depends on persons number, such as: sending and receiving, pre processing and more.

fps for different image size

Here we take images of different sizes and check how it affects fps:



As we can see, fps increase as image size gets smaller. The problem is, that there is a trade off between working very fast on small images to our mission of making good detections. Fps in the range of 10-11 was reached by images size of 300-400, that were still big enough for detection, so we decided to work with images size of 360,480.

Conclusion

Our client-server application is a great tool in many ways, for example, for salesmen seeking to sell their products.

We improved the DeepFace framework by merging it with YOLO. This merge improved the fps we measured by approximately 25%. This was the most challenging task we faced since we had to get deeply into the frameworks code and update it.

Furthermore, we were able to detect multiple people's emotions in a video/photo instead of just one person's. Also, we were able to improve the time it takes to pass the image from the client to the server by resizing the image while retaining the same results.

Future work

- Run the server side on GPU – GPU is used for accelerating computational processes and parallelization. Because we run the image into convolutional network, which considered as heavy computational process, using a GPU can save us a lot of time, and can be splitted between some computational resources instead using only one. Saving time is very important when dealing with real time applications.
- Change the way we detect face – detect once and than track the bounding box. Changing the way we detect faces can save us time and be more efficient. Instead of doing face detection each time, we doing it for the first time and then we just track the pixels of the face that we found.
- User friendly App that uses our platform – improve our UI and app visuality.

Bibliography

1. YOLOv3: An Incremental Improvement
   https://pjreddie.com/media/files/papers/YOLOv3.pdf\
2. YOLO-face: a real-time face detector
   https://www.researchgate.net/publication/339888559_YOLO-face_a_real-time_face_detector
3. DeepFace
   https://www.cs.toronto.edu/~ranzato/publications/taigman_cvpr14.pdf
4. Django
   https://www.djangoproject.com/