

Facial Expression Detection

Authors

André Mourato 84745 Nuno Félix 80330

Abstract - This report describes the development and demonstration of the Facial Expression Detection project chosen by our group for the Visual Computation subject.

I. INTRODUCTION

The main goal of this project was to learn and create a program that detects facial expressions. We trained a neural network to recognize different emotions from various inputs: an image, a video or a webcam.

II. DEMONSTRATION

As stated above, our project allows the user to identify facial expressions through the webcam, video or image support. We will begin this report by presenting the results that we have achieved and then we will explain how that was possible, by giving an overview explanation of the process of training a neural network, since the neural network is beyond the scope of this project. For the neural network we used software from Tensorflow. A demonstration video can be found here (https://www.youtube.com/watch?v=nyCssLO0_6g).

A. Webcam

In **Webcam** mode, the feed from your web camera will be sent to the application which will identify the expressions of people in real time.

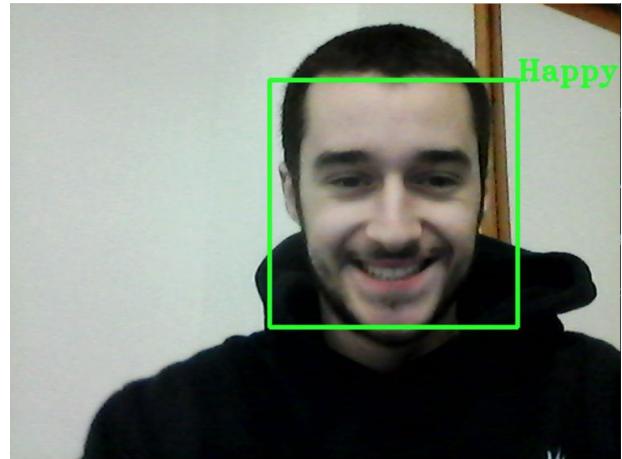


Fig. 1 - Happy emotion detection

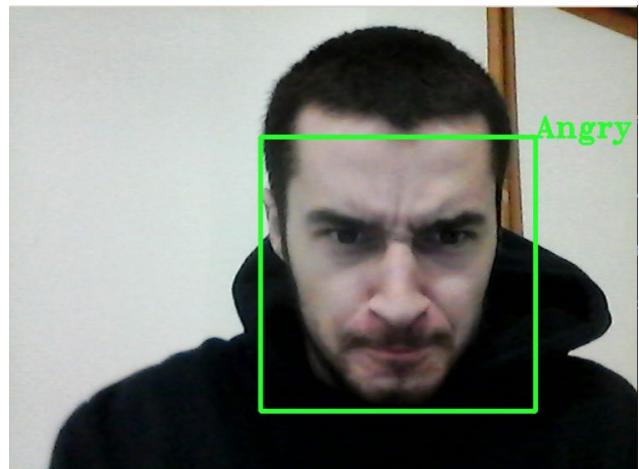


Fig. 2 - Sad expression detection

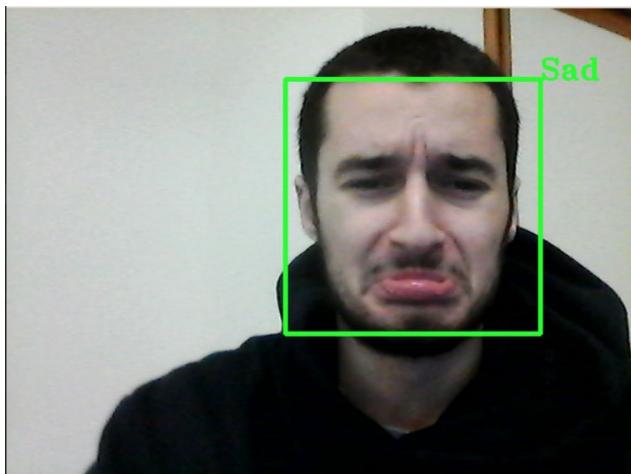


Fig. 2 - Angry emotion detection

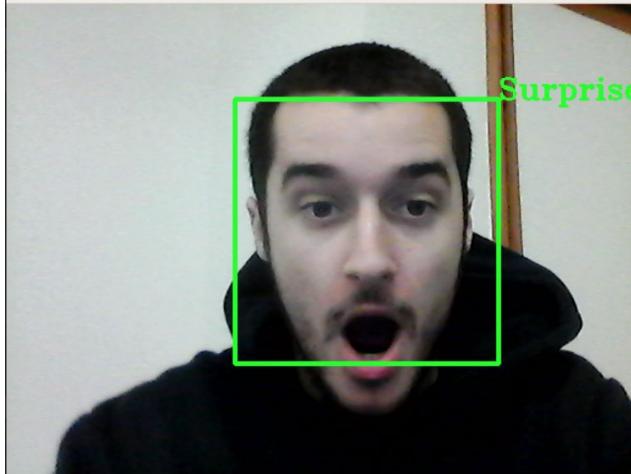


Fig. 3 - Surprise emotion detection

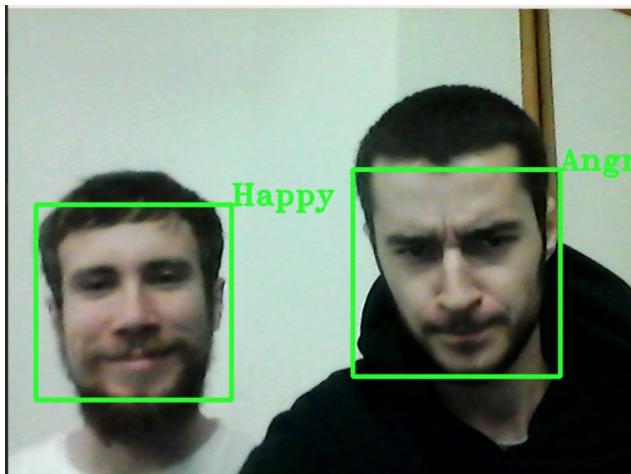


Fig. 4 - Dual face expression detection

B. Video

In **Video** mode a video must be provided as argument. The facial expressions will be detected and presented to the user in real time.

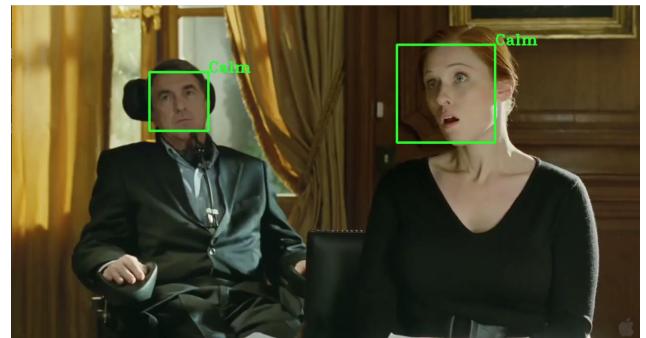


Fig. 5 - Dual face expression detection in video

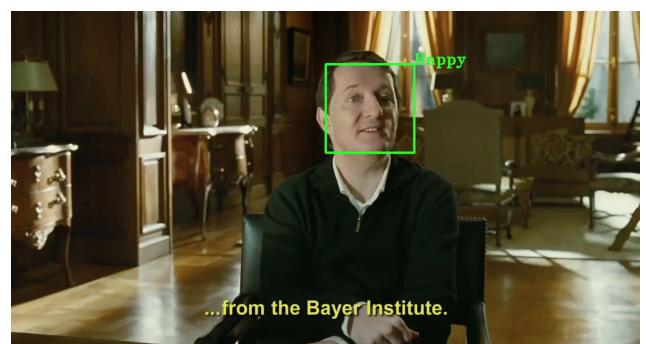


Fig. 6 - Happy expression detected in video

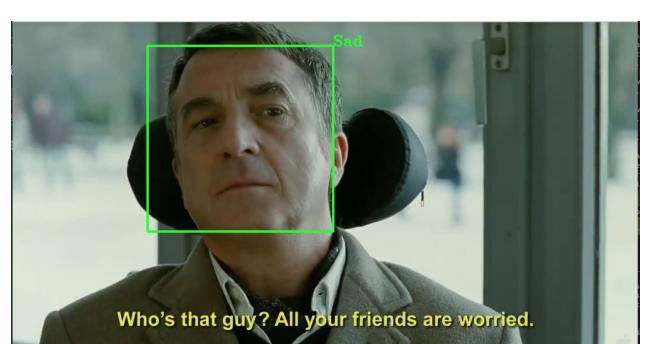


Fig. 7 - Sad expression detected in video

C. Image

The application also accepts an image file as input that will be processed and shown to the user.

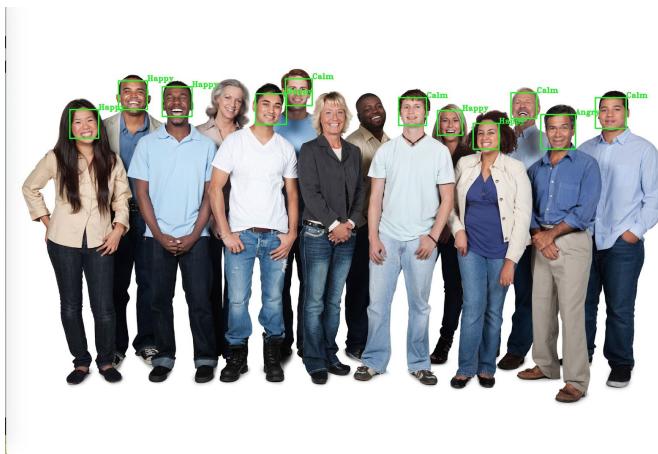


Fig. 8 - Detecting multi faces expressions in an image

D. Training network

The detection models are represented in the *retrained_graph.pb* and the *retrained_labels.txt* file, which is located in */src/*. The existing files correspond to the version our group trained, but it is possible to train these models. This process will be explained in the DEPLOYMENT chapter of our report. To train the models, a dataset is required. This dataset should contain photos of different faces expressing different emotions. We observed that the quality of the models improved when more diverse datasets were used (more faces were tested). Our data set is located in the */src/image* and is divided into various subfolders. Each of these subfolders contain images of a different emotion.

III. DEVELOPMENT

A. Loading Webcam and Video

We can detect facial expressions using the webcam, a video or image. This is achieved by using OpenCV VideoCapture class for video capturing from video files or cameras and the imread function for the images.

B. Import and use of OpenCV Haar Cascades

We used a premade Haar Cascade face detection model, that is widely used, and can be found in (https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_frontalface_alt.xml) called “Front Face Alt” classifier for detecting the presence of the front part of a face. We import this classifier in facialrecognition.py

```
classifier = cv2.CascadeClassifier('haarcascade_frontalface_alt.xml')
```

C. Loading a model and retrain a Deep Learning Algorithm

Image classification models have millions of parameters. Training them from scratch requires a lot of labeled training data and a lot of computing power. Transfer learning is a technique that shortcuts much of this by taking a piece of a model that has already been trained on a related task and reusing it in a new model.

We used and modified a script, from tensorflow, to load pretrained models, and it can be found here (https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/label_image/label_image.py).

For retraining the model we also used an algorithm, developed by Tensorflow, that can be found here (https://github.com/tensorflow/hub/blob/master/examples/image_retraining/retrain.py).

This process can be described by the following steps:

- We created a directory called images, in this directory we created subdirectories for each emotion we want to identify, for example, happy, calm, sad, angry, surprised.
- In these directories, we insert images of the different emotions (images of happy faces in the Happy subdirectory, for example) or by using the script utils.py to capture an image from the webcam and inserting the image in the respective folder automatically. The more diverse the dataset, the better the quality of the detection models.
- After the previous steps, we run the retrain.py script. This process can take around 5 minutes to complete, since it has to train the models with all the images.

D. Improving GPU performance

When testing we noticed that videos and camera capture were running at 4 fps when detecting facial expressions. This happened because the CPU was being used and it is not able to perform so many calculations in such a short amount of time. We solved this by using a GTX 1060 and a GTX 950m GPU, that supported CUDA, to perform the

heavy computation, installing Tensorflow GPU version and CUDA software and drivers. We noticed some improvements but the results were still not satisfactory, since we were getting 10 fps and the GPUs are capable of more than that framerate. After some research we discovered that the original tensorflow script was developed to analyse only 1 image and not a video, so it was creating many tf.Session objects unnecessarily. We solved this problem by initializing a single tf.Session object in the beginning of the application and the framerate went from 10 fps to 30 fps and even 40 fps in some cases. Overall, we saw a drastic improvement in performance, almost 7.5x better since the non-optimized version of the application.

IV. DEPLOYMENT

Throughout the development of this project the group used Git as a version control system. More information can be found in the repository of the project in GitHub: <https://github.com/andremourato/FacialExpressionRecognition>

To deploy our project, some prerequisites need to be installed:

```
pip3 install -r requirements.txt
```

This includes opencv-python and Tensorflow CPU version. For the GPU optimized version you should have a GPU that supports CUDA and you must install Tensorflow GPU. This can be done by running the following commands:

```
pip3 uninstall tensorflow==1.14.0rc0
pip3 install tensorflow-gpu==1.14.0rc0
```

You'll also need to install CUDA, a parallel computing platform made by NVIDIA and their drivers, the installation guide can be found in the following link (<https://www.tensorflow.org/install/gpu>).

For running the app with the webcam you must connect your web camera to the computer and run the following command:

```
python3 facialrecognition.py
```

For running the app with a video or image

```
python3 facialrecognition.py
[video-or-image-path]
```

You can train the detection model by using your own face by running the utils.py script and passing as arguments 'train_emotion_from_webcam', your name followed by the expression you want to train.

```
python3 utils.py train_emotion_from_webcam
felix Angry
```

For retraining the model with the newer models and expressions, run:

```
python3 retrain.py
--output_graph=retrained_graph.pb
--output_labels=retrained_labels.txt
--architecture=MobileNet_1.0_224
--image_dir=images
```

NOTES

This project was developed using python, because the deep learning algorithm we were trying to use was only available in Tensorflow Python version.

REFERENCES

- [1] <https://www.tensorflow.org/install/gpu>
- [2] https://docs.opencv.org/master/d6/d00/tutorial_py_root.html
- [3] https://www.tensorflow.org/versions/r1.14/api_docs/python/tf
- [4] https://github.com/tensorflow/hub/blob/master/examples/image_retraining/retrain.py
- [5] https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/label_image/label_image.py