TUT Embedded Smile Detector

NVIDIA Jetson Developer Challenge

Feb 2018

Tampere University of Technology

- Pedram Ghazi
- Saboktakin Hayati
- Heikki Huttunen

Table of contents

Table of Figures

1. Introduction

The original Idea was to extend facial expression recognition to people's lives in order to prevent threatening behaviours .
Here we describe some possible use cases:

I. Creating a system which is able to detect children's mood enables us to deploy a framework for detecting a range of disease in children, e.g. if a child stares for a long time during his break, It could be the sign of Epilepsy, Autism, Child abusement , Family problems, etc.

II. Increasing people efficiency at work could be achieved by detecting their mood at work and taking proper actions to prevent spread of bad feelings as much as possible.

III. In an interview situation, the interviewee can be monitored to detect his moods and level of stress during the interview. So, the interviewers can act accordingly to interviewee’s feelings.

2. Implementation

2.1 Implementation

For implementing this system on an embedded device namely Jetson TX2 we have used multithreading since computational overheads from different modules of the system would cause inconsistency in the execution flow of the application. The main thread is responsible for handling the communication between different threads and modules. The general flow of the program could be explained as this: firstly we are grabbing frames in the main thread and share them between all threads using a data structure which all threads have access to it. Secondly, we are detecting faces in the captured frames and assigning coordinates of the detected faces to the shared data structure. Then the main thread is retrieving the prediction results of smiles in those coordinates form the smile detector thread-which is making use of a deep learning architecture namely mobilenet(Howard, A.G. 2017.)-. In the final step, we sketch the frame with the prediction result added to it.
Handling multiple faces for smile detection was carried out by accessing coordinates of rectangles of detected faces in a frame by the smile detector thread and assigning smile probabilities to each rectangle. So while sketching the frame we will be able to draw the rectangle and the prediction result alongside each other.
The communication between different threads are asynchronous, so they do not need to be available at the same time. Since all threads are running almost at the same pace, we have a stable system which is able to draw approximately 18 frames ps.

2.2 Workflow

The current workflow of the application could be described as in this sequence diagram.

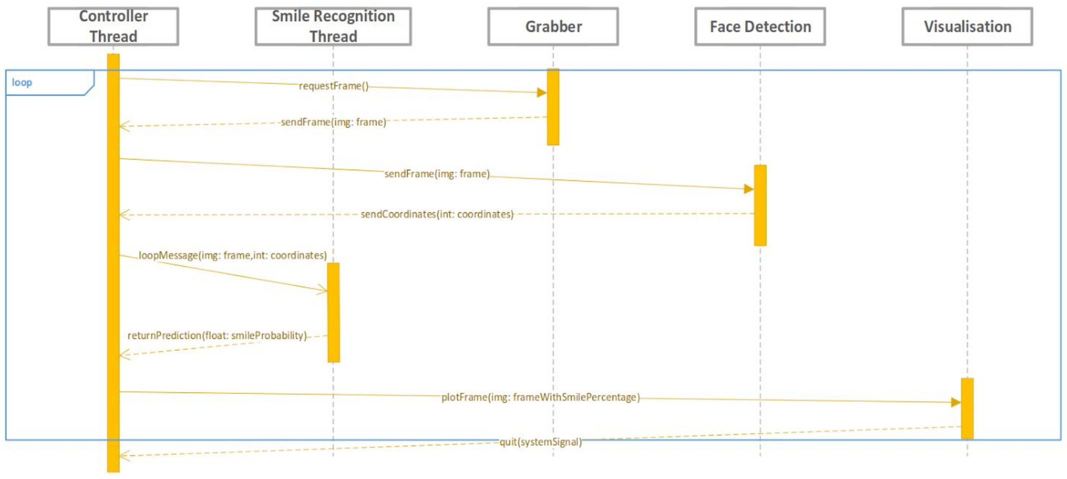

Controller Thread
Smile Recognition Thread
Grabber
Face Detection
Visualisation
loop
requestFrame()
sendFrame(img: frame)
sendFrame(img: frame)
sendCoordinates(int: coordinates)
loopMessage(img: frame,int: coordinates)
returnPrediction(float: smileProbability)
plotFrame(img: frameWithSmilePercentage)
quit(systemSignal)

Figure 2.1: Sequence diagram

2.3 What did go wrong?

We wanted to have a real time plot of diagrams depicting smile probabilities, We implemented such a system which was working on a PC for one face but when we tested that on TX2 the whole system froze due to lack of computation power in its CPU.

2.4 Environment and libraries

TX2 jetson owner is Tampere university of technology, signal processing lab and they lent us the device for this project. We also used Logitech webcam c920 as a camera.

Tools(libraries, apps, softwares)	**Version**	**Explanation**
Ubuntu	16.04 LTS	Operating system-64 bit.
Python	2.7.12	Programming language.
OpenCV	2.4.13.1	Library which is mainly use for computer vision.
Keras	2.1.1	A neural network library.
Tensorflow	1.3.0	A library for dataflow programming across a range of tasks including machine learning.
Numpy	1.11.0	A library for scientific computing in python.

3. Conclusion

In conclusion, although we were not able to develop the project further due to lack of resources, we gained the hands-on experience of implementing an idea from the scratch to a working application.
As future works, we have plans as follows:
- Pretrain model with other datasets to make it more robust.
- Detecting other facial expressions than smiling.
- Consider experts' ideas to include diverse domain knowledge to make this application more practical.
- Improve thread programming.

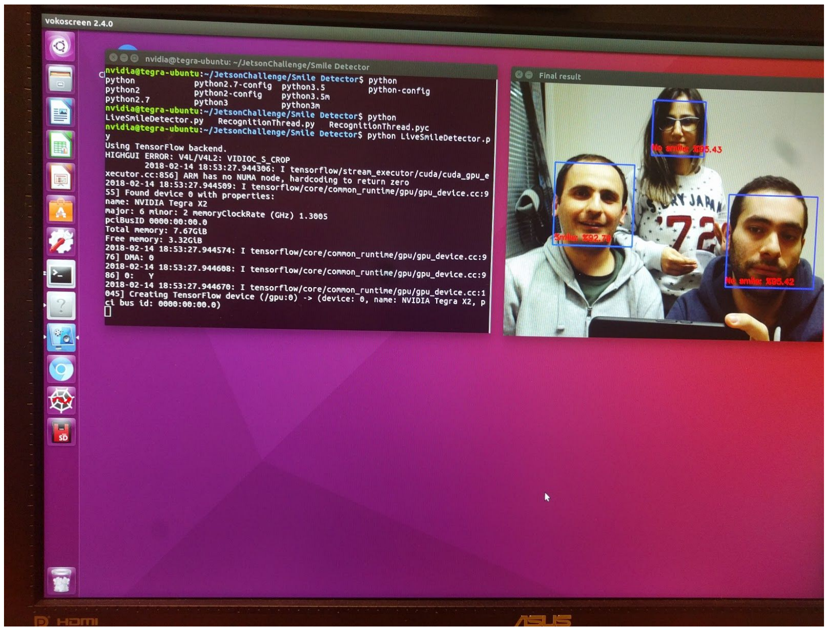
In the end, we would like to say thanks to our university teacher Heikki Huttunen for his support during the project.

4. References

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. and Adam, H., 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

5. Annexes

5.1 Images



```
nvidia@tegra-ubuntu:~/JetsonChallenge/Smile Detector$ python
python              python2.7-config   python3.5          python-config
python2             python2-config     python3.5m
python2.7           python3            python3m
nvidia@tegra-ubuntu:~/JetsonChallenge/Smile Detector$ python
LiveSmileDetector.py   RecognitionThread.py   RecognitionThread.pyc
nvidia@tegra-ubuntu:~/JetsonChallenge/Smile Detector$ python LiveSmileDetector.p
y
Using TensorFlow backend.
HIGHGUI ERROR: V4L/V4L2: VIDIOC_S_CROP
        2018-02-14 18:53:27.944306: I tensorflow/stream_executor/cuda/cuda_gpu_e
xecutor.cc:856] ARM has no NUMA node, hardcoding to return zero
2018-02-14 18:53:27.944509: I tensorflow/core/common_runtime/gpu/gpu_device.cc:9
55] Found device 0 with properties:
name: NVIDIA Tegra X2
major: 6 minor: 2 memoryClockRate (GHz) 1.3005
pciBusID 0000:00:00.0
Total memory: 7.67GiB
Free memory: 3.32GiB
2018-02-14 18:53:27.944574: I tensorflow/core/common_runtime/gpu/gpu_device.cc:9
76] DMA: 0
2018-02-14 18:53:27.944608: I tensorflow/core/common_runtime/gpu/gpu_device.cc:9
86] 0:   Y
2018-02-14 18:53:27.944670: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1
045] Creating TensorFlow device (/gpu:0) -> (device: 0, name: NVIDIA Tegra X2, p
ci bus id: 0000:00:00.0)
```

Figure 5.1: Creating the demo

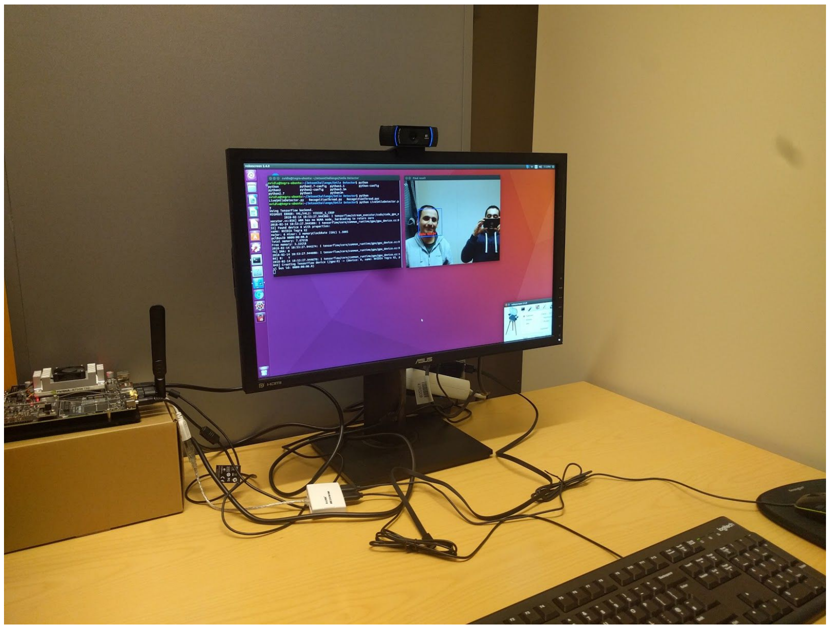


Figure 5.2: Hardware

5.2 links

Link to the public repository: https://github.com/alitakin/JetsonChallenge

Link to the demo: https://www.youtube.com/watch?v=4JGatQOchFo