Smile Detection

Valentin DE BALTHASAR DE GACHEO
Dept. of Computer Science and Information Engineering
National Taiwan University of Science and Technology (NTUST)
Taipei, Taiwan
valentindebalthasar@gmail.com

Sabina SERRA
Dept. of Computer Science and Information Engineering
Linköping University (LiU)
Linköping, Sweden
Sabse455@student.liu.se

Roch MOREAU
Dept. of Computer Science and Information Engineering
Ecole Internationale des Sciences du Traitement de l'Information (EISTI)
Cergy, France
moreauroch@eisti.eu

Paul MENARD
Dept. of Computer Science and Information Engineering
University of Technology of Compiegne (UTC)
Compiegne, France
menard5@hotmail.fr

Keywords—Artificial Intelligence, Machine learning, Deep learning, Computer vision, Image detection, Face detection, Image classification, Sentiment detection.

I. Introduction

The goal of this project is to create an embedded facial emotion detection and classification running on android devices. The project is based on a Deep Learning model, trained with a public dataset available on Kaggle's website.

The first model developed is inspired by a model that is comparatively small and achieves almost state-of-art performance of classifying emotion on this data-set. The architecture was proposed by Octavio Arragia et al. [1]. Finally, a second model was developed, inspired by YoloTiny model[2].

The first approach was to use qualcomm SNPE to run and deploy the model on a mobile device, but in the end Tensorflow Lite was used instead. With Tensorflow Lite the model was able to successfully run on Android devices and *The Deeply Boring App* was created. The *Deeply Boring App* detects when people smile with an accuracy around 90%.

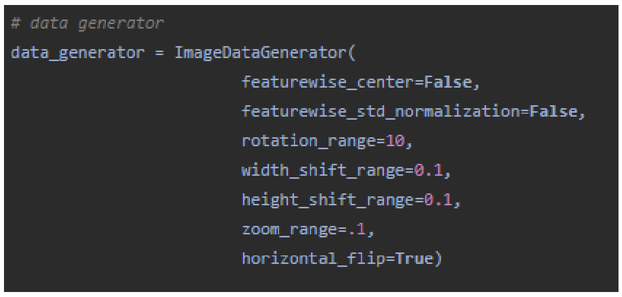
II. Method

A. Material Used

1) Dataset

The dataset used is FER2013 and is available online [3].

Data augmentation is performed to increase the number of instances in the data and therefore improve accuracy. The details of the data augmentation carried on in this project are presented below.



```
# data generator
data_generator = ImageDataGenerator(
                        featurewise_center=False,
                        featurewise_std_normalization=False,
                        rotation_range=10,
                        width_shift_range=0.1,
                        height_shift_range=0.1,
                        zoom_range=.1,
                        horizontal_flip=True)
```

Fig.1 Data Augmentation

2) Data Structure

The dataset is a single "*.csv*" file containing three columns:

1. "emotion": Integer from 0 to 6, describing the emotion expressed which are used as labels.
2. "pixels": Containing a list of integers representing an image in bitmap format.
3. "Usage": Can be either "training" or "PublicTest", but which are not used in this project to define the training and testing sets.

3) *Models Used*

Two models are developed during this projects, both based on state-of-the-art results in this field.

Both models are using the same input/output layers' set described in the picture below.

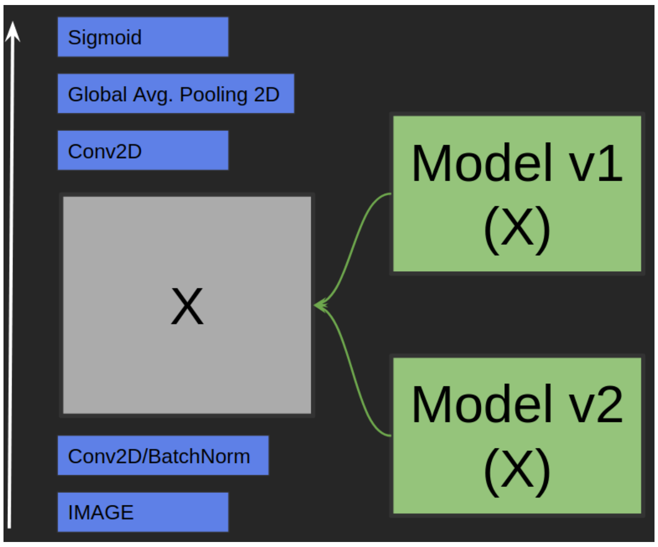


Fig.2: Model base

The first model developed in this project is inspired by previous findings found in the literature in this field [1]. An accuracy of 66% is achieved with the latter model. However, as only two classes are considered in this model (happy / others) the last layer is changed to be a sigmoid function.

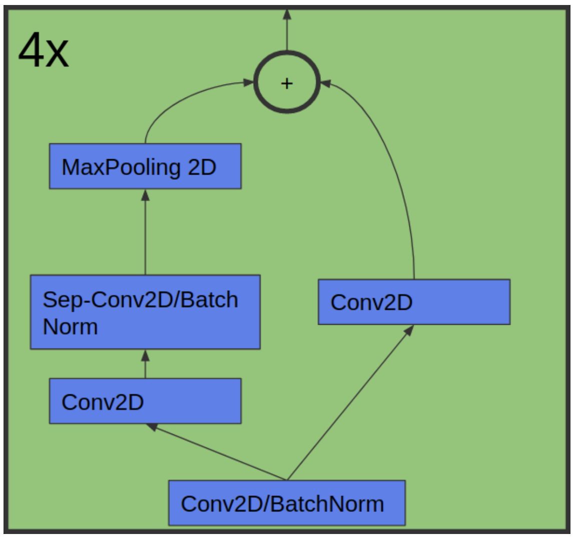


Fig.3: Model V1

A second model is designed to challenge the first one. The latter is inspired by Yolo Tiny model[2], as it is known for providing good accuracy, but also making quick inferences, which is required when running on an embedded system with limited computational power.

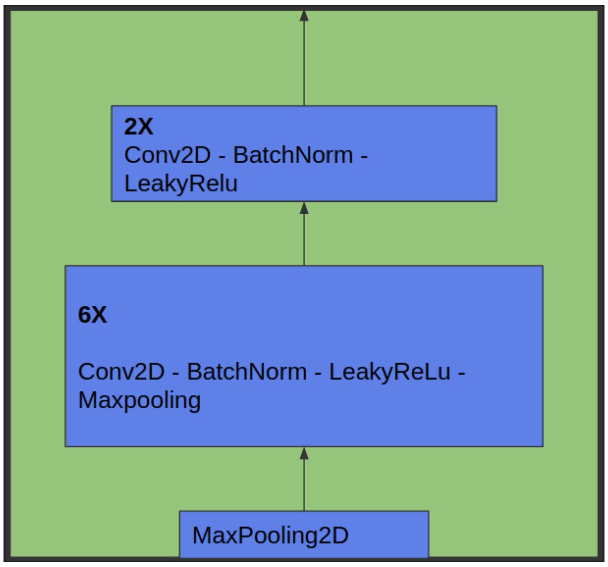

2X
Conv2D - BatchNorm - LeakyRelu
6X
Conv2D - BatchNorm - LeakyReLu - Maxpooling
MaxPooling2D

Fig.4: Model V2

4) *Hardware specificities*

Below, please find the specificities of the hardwares used in the experiments:

a) Computer

- RAM: 12Go
- Processor: i7-5700HQ @2.7GHz 8CPU
- Operating system: Windows 10 Pro

b) Embedded Device

- RAM: 3Go
- Processor: ARM Cortex-A53 - 1.9 GHz (8 CPU)
- Operating system: Android 8

B. *Implementation Pipelines & Experiments*

1) *Pipelines Figures*

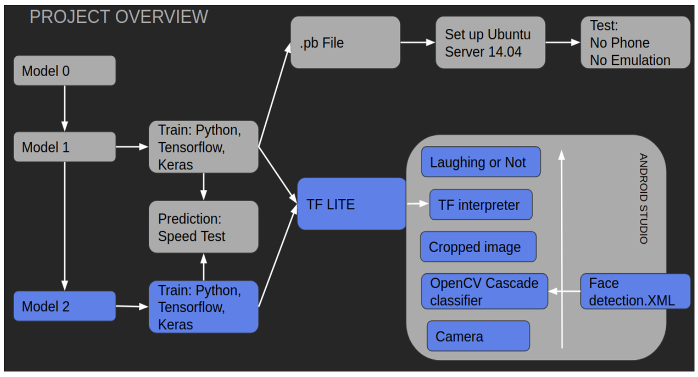


Fig.5: Method Pipelines

2) Pipelines Description

First an already existing state-of-the-art model is used to test the current emotion recognition. Changes to the first model are then made after concluding that everything worked on our devices.

Since Qualcomm only supports Tensorflow, our first model is converted from hdf5 to pb. Qualcomm only supports Ubuntu 14.04 so a server is set up to run the model. At this stage it is discovered that there are no emulators in Android studio for phones supported by Qualcomm. SNPE SDK is still used to ensure our model pass the build, but we cannot benchmark the model on a real qualcomm CPU. Since neither a phone nor a emulator are provided, Tensorflow Lite is used for the rest of the project.

To use Tensorflow Lite our Keras model is converted to the Tensorflow lite format (.tflite). Our app is based on the Tensorflow example "Object Classification" [4] but is changed to fit emotion recognition instead of object classification. Since our model is trained on frontal-face pictures, face detection is performed before the emotion recognition. The haar cascade classifier from OpenCV is used to perform the face detection. Emotion recognition is then performed on the cropped image after it has been converted to gray scale.

After a performance test between the two models using the app, the second one is chosen to replace the first one.

C. *Issues Faced*

III. Results

A. *Predictions on Computer*

B. *Predictions on Embedded Device*

C.

1) Comparison

Models	Results	
	Validation Accuracy	***Inference Time [180ms/image]***
Model_v1	0.91	10.7
Model_v2	0.92	10.6

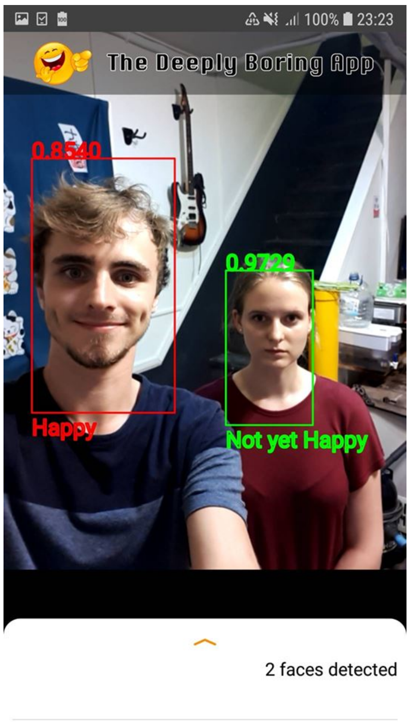
Fig.6: Model V1

D. *Models' Results on Embedded Device*

Models	Results
	Inference Time [ms/image]
Model v2	180

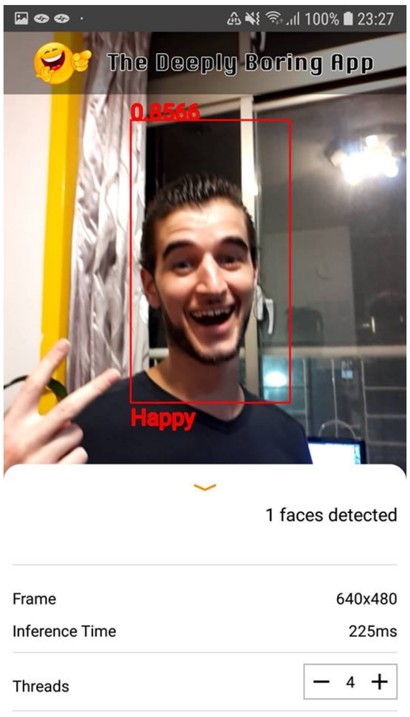
Fig.7: Model V1

Below, a screenshot of the program layout:



2 faces detected

Fig.8: Model V1



1 faces detected

Frame 640x480

Inference Time 225ms

Threads 4

Fig.9: Model V1

IV. Discussion

This project could be used in future apps as a new interaction method with the user. A mobile app able to react to users' emotions could be developed.

Below, some examples of future usage of this project:

- Data collection: Gathering users' reactions of displayed content (like movie, ads, articles). This data could be used to tailor content to a users liking. If a user reacts badly to an ad, the provider can for example stop showing it to him/her.
- Instant interaction: A game like “You laugh you lose” where the user looks at funny videos and if he laughs, he loses.

In addition to this, the inference speed of the second model could be improved by simplifying the complexity of the model, and by tuning hyperparameters. In addition to that, Yolo-LITE could be used, as it is a real-time object detection algorithm optimized for non-CPU computers [5].

V. Conclusion

Two models were developed in this project in order to perform sentiments classification. Thanks to the high validation accuracy (92%) and the responsiveness of our model (10.6ms/image), it could run using an android system with limited computational power.

Further improvements expressed in the discussion could participate in the implementation of this model into an app that could run with even more limited computations and higher inference speed.

References

[1] "Real-time Convolutional Neural Networks for Emotion and Gender Classification." Octavio Arriaga, Paul G. Ploger, Matias Valdenegro, 2017, https://arxiv.org/pdf/1710.07557.pdf

[2] J. Redmon, and Farhadi, “A. YOLO9000: better, faster, stronger”, arXiv, 2017, https://arxiv.org/pdf/1612.08242.pdf

[3] “Challenges in Representation Learning: A report on three machine learning contests” Ian J. Goodfellow et al., 2013, https://arxiv.org/pdf/1307.0414.pdf

[4] https://github.com/tensorflow/examples/tree/master/lite/examples/image_classification/android

[5] “YOLO-LITE: A Real-Time Object Detection Algorithm Optimized for Non-GPU Computers”, Rachel Huang, Jonathan Pedoeem, Cuixian Chen, 2018, https://arxiv.org/pdf/1811.05588.pdf