Eyes direction recognition in simulator-generated faces

Mele Alessandro[1] and Traini Davide[2]

[1]Master's Degree students in Computer Science and Automation engineering, Università Politecnica delle Marche, Ancona, Italy

***Abstract*— The aim of the following paper is to design and develop neural networks for classification and regression of eyes direction of drivers. We have created a custom dataset formed by images acquired through a simulator on web and then trained a Mobile Net V2 to classify nine dials. Dataset is composed in total of 5.386 images, all named as *horPerc_verPerc.png* , where *Perc(s)* are normalized than the maximum and minimum values of the cursor positions, such a way as to make easier the regression task. Results are very successfully on validation test, but for now, they are not satisfactory on other external set of images, including video game and real faces. Our project is available at the following GitHub repository.**

I. Introduction

The increasing development of safety in vehicles has made the driver the weak link in the chain. To help devolved, tired or sleepy drivers, vehicles are being produced with driver assistance systems: these, known as Advanced Driver Assistance Systems (ADAS) or DAS, make extensive use of Artificial Vision. Current solutions propose: Intelligent Transportation Systems, to ensure safe navigation, increase efficiency and avoid traffic; assisted/autonomous driving, to replace, partially and/or not, the driver with the vehicle itself; Advanced Driver Assistance Systems, to monitor the movement and flow of vehicles along the road network. Several efforts to develop active safety systems have been reported in the literature for reducing the number of automobile accidents due to reduced vigilance. The maximum detection accuracy is obtained with measurement of physiological conditions through intrusive techniques like brain waves, heart rate, and pulse rate . Less intrusive techniques like eyelid movement, gaze detection or head movement monitoring techniques with head-mounted devices as eye tracker or special contact lens also deliver good results. These techniques can also be used to detect situations where the driver is not paying attention to the road, or is distracted by external events.

II. State of the Art

A. Dataset

Recently, have been collected several datasets where subjects are asked to look at pre-defined targets on the screen of a mobile device, with the aim of introducing greater variation in lighting and appearance. These datasets are suitable for situations where a subject is directly facing a screen or mobile device; but typically many of these datasets don't contain eye gaze labels.

B. Method

In the last five years conventional approaches, such as k-Nearest Neighbors or Random Forests, have been outperformed by neural network. Krafka et al. [1] introduced a CNN which estimates the gaze by combining the left eye, right eye and face images, with a face grid, providing the network with information about the location and size of the head in the original image. Deng and Zhu [2] suggested a two-step training, where two CNN are trained separately, one on eyes and one on head images, then jointly fine-tuned with a geometrically constrained "gaze transform layer".

III. Dataset creation

In this section we present each step of the dataset creation: for our purposes, we have used a face simulator provided by Western University [3].
The first step has been acquiring images from the simulator: for this, we have developed a Node.JS [4] bot that automatically moves mouse cursor and takes screenshots on the web page.
For the classification task, we have made filters for nine classes.

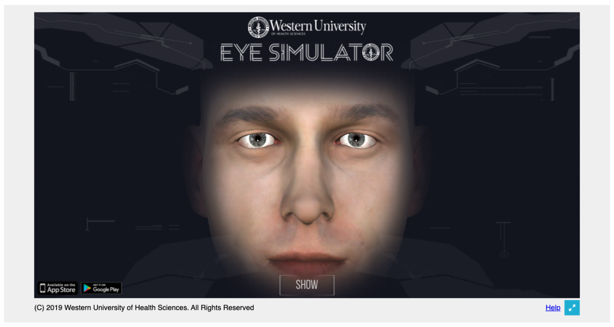
Western University
EYE SIMULATOR
SHOW
(C) 2019 Western University of Health Sciences. All Rights Reserved
Help

Fig. 1. Eye simulator from Western University

The second step has been data augmentation: we have made rotations (by ± 5 degrees) and prospective transformations (with a delta factor of 80) for variability on points of view of the camera and applied color curve for variability on color skin.
Rotations and transformations have been done through OpenCV

[5] in Python, while color curve has been applied through BIMP [6], a GIMP [7] plugin.

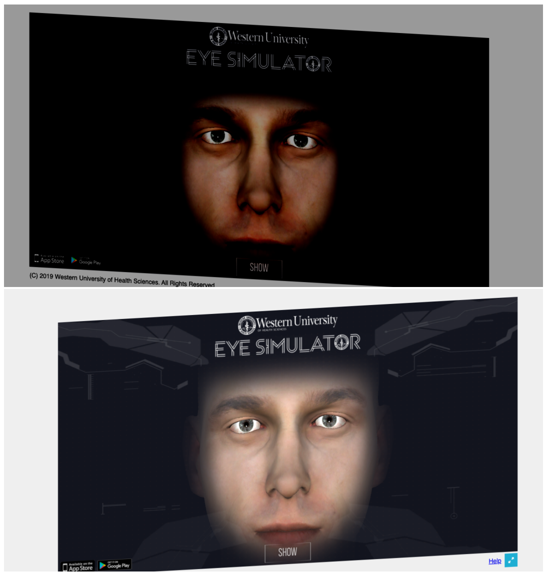

Fig. 2. Example warping and color curve applied on a image.

The last step for dataset has been to apply Face Mesh [8] to all previous images.

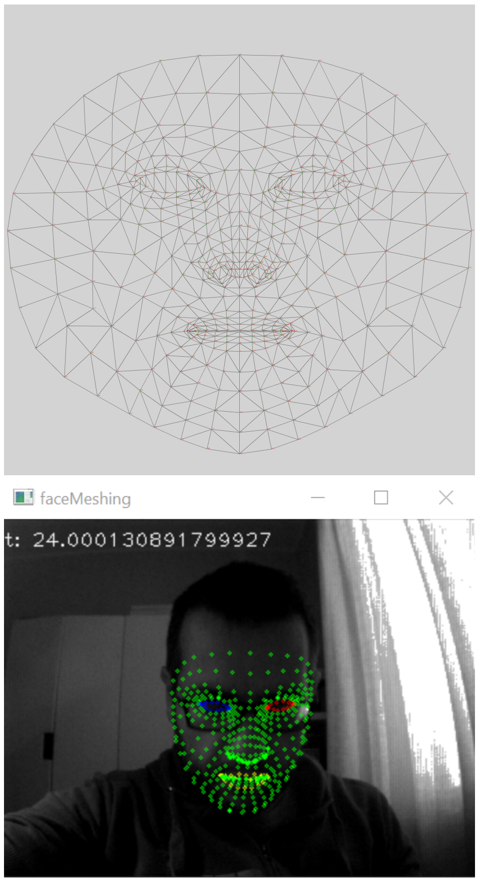

Fig. 3. 468 3D face landmarks mask and example of application.

This solution has been applied for cutting automatically eyes in previous images thanks to bounding boxes derived by Face Mesh processing. Then, we have resized all images to square size (100x100 pixels) to ensure the correct input to the neural network. In the end, our dataset is exclusively formed by images such as the following:

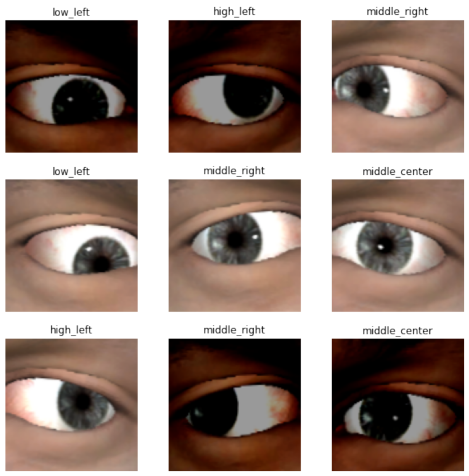

Fig. 4. Eyes cropped and resized by Face Mesh.

IV. Model training

For the classification and regression tasks we have used Mobile Net V2, because it is one of the fastest networks so, it fits the purpose of the paper.
The dataset, composed by 5.376 eyes images, was divided into 80% training, 15%validation and 5% test; each batch contains 64 images, with a total of 68 batches.

A. *Classification*

The complete model consists on:

- One layer for the normalization of the input;
- Layers of the Mobile Net;
- One dense layer with nine neurons activated with a *Soft-Max* function.

The training has been divided in two phases:

- in the first, the classification layer has been trained in 20 epochs, with:
  - *Sparse Categorical Cross Entropy* as loss function;
  - *Learning rate* equal to 0.001;
  - Adam as optimizer;
  - Dropout equal to 0.3.
- in the second phase:
  - All the layers after the 80 has been unfreezed;
  - Training has continued again for 20 epochs, with a *learning rate* equal to 0.0001.

B. *Regression*

Starting from the 5.376 images, it has been created a *dataframe* formed by *image_path, perc_hor, perc_ver* which has been passed to the function *flow_from_dataframe* to create the regression dataset.
The complete model consists of:

- One layer for the normalization of the input;
- Layers of the Mobile Net;
- One dense layer with two neurons without an activation function.

The training has been divided in two phases:

- in the first, the regression layer has been trained in 20 epochs, with:
  - *Mean squared error* as loss function;
  - *Learning rate* equal to 0.01;
  - Adam as optimizer;
  - Dropout equal to 0.3.
- in the second phase:
  - All the layers after the 80 has been unfreezed;
  - Training has continued again for 20 epochs, with a *Learning rate* equal to 0.0001.

V. Results

Both classification and regression models has a good accuracy on synthetic test sets, but not on test datasets formed by real faces.
From the confusion matrix on synthetic test, we show as on a batch formed by 64 items, the prediction fails only on two ones.

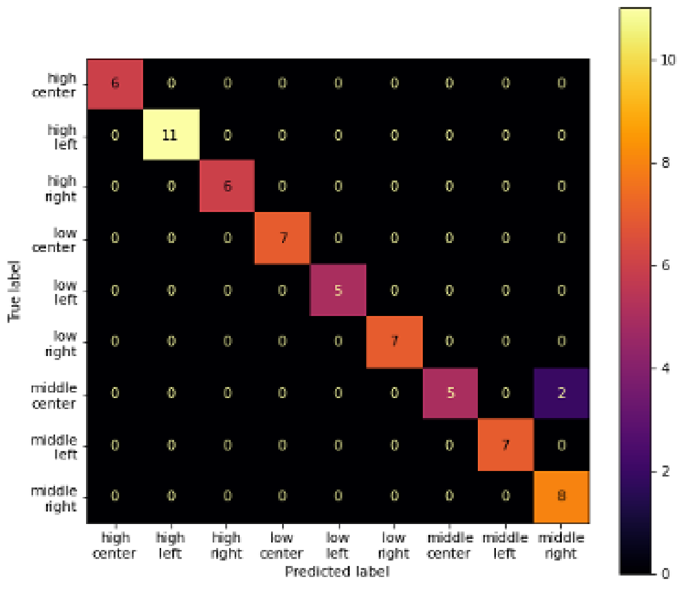

Fig. 5. Confusion matrix.

For the regression, have been used three metrics: minimum squared error (MSE), minimum absolute error (MAE) and minimum absolute percents error (MAPE) and respective values are 0.003, 0.0423 and 39.9776 on synthetic test. Using real images the model was not able to generalize, so we tried to resolve with three methods:

- Train one model for left and another for right eye;
- Train the model with only *middle_left* and *middle_right* classes;
- Apply a cost matrix, to avoid unbalanced class problems.

The above listed methods were not sufficiently able to mitigate the generalization issue. A more complete data-set with synthetic and real images is a good candidate for that.

VI. Conclusions

In this paper we exposed the design and develop of neural networks for classification and regression of eyes direction of drivers, and we showed as both are really accurate on synthetic dataset, but have poor performances on real world data. A further approach would be to:

- Train the network on both synthetic and real data.
- Evaluation of other architectural networks;

References

[1] Kyle Krafka, Aditya Khosla, Petr Kellnhofer, Harini Kannan, Suchendra Bhandarkar, Wojciech Matusik, and Antonio Torralba. "Eye Tracking for Everyone". In: (2016), pp. 2176–2184. DOI: 10.1109/CVPR.2016.239.

[2] Haoping Deng and Wangjiang Zhu. "Monocular Free-Head 3D Gaze Tracking with Deep Learning and Geometry Constraints". In: (2017), pp. 3162–3171. DOI: 10.1109/ICCV.2017.341.

[3] *Western University Eye Simulator*. URL: https://edtech.westernu.edu/3D-eye-movement-simulator/.

[4] *Node.JS: A JavaScript runtime built on Chrome's V8 JavaScript engine*. URL: https://nodejs.org/it/.

[5] *OpenCV: A library for Computer Vision tasks*. URL: https://opencv.org.

[6] *BIMP: An easy interface and a lot of batch functions to apply on a set of images*. URL: https://alessandrofrancesconi.it/projects/bimp/.

[7] *GIMP: A cross-platform image editor*. URL: https://www.gimp.org.

[8] *Face Mesh: a solution for estimating face landmarks*. URL: https://google.github.io/mediapipe/solutions/face_mesh.html.