Project Documentation

Driver Distraction Detection Using Deep Learning and Raspberry Pi

Submitted By

Hanan Alharbi

Ahmad Al-kaf

Neaam Hariri

Mentor:

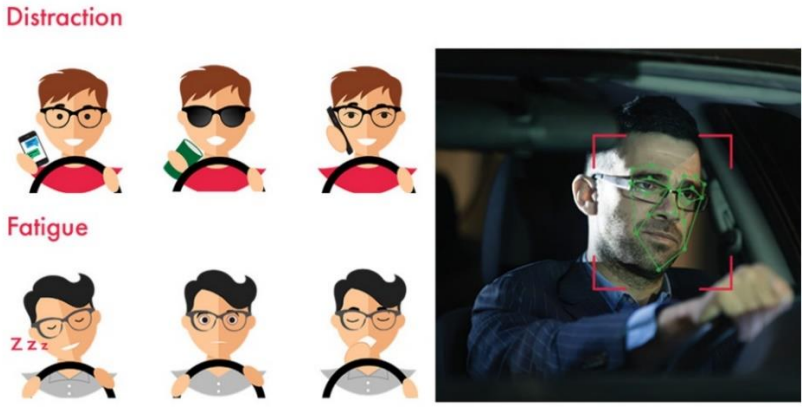
Sarah Khayyat

2019

Overview:

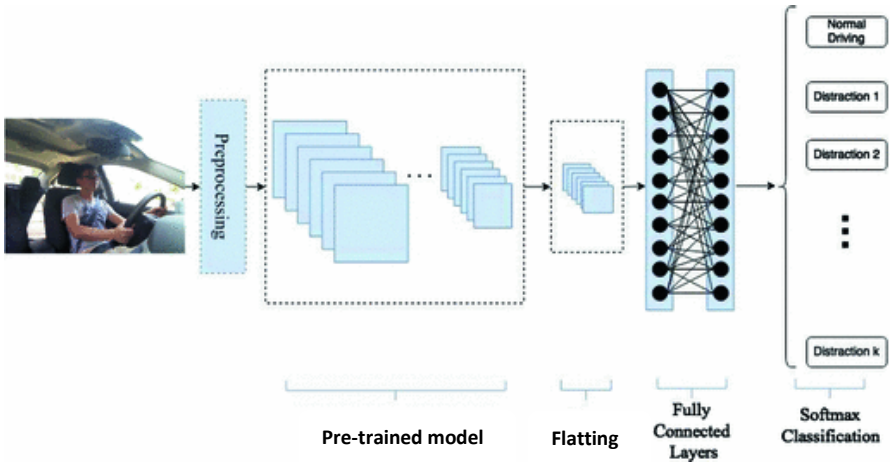
95% of fatal vehicle accidents are caused by human faults*, most of which are from the driver being distracted. The distraction while driving can happen in many ways, whether the driver call or text or just holding their phones, drinking, smoking, reaching behind, or even doing their hair or makeup. All these cases can cause the driver to lose their focus on the road.

So, we will design a model that works on deep learning properties. We will train the model to detect such behaviours, and to alert the driver. This would reduce the risks involved.



Proposed solution:

Driver Distraction Detection system uses deep learning to detect and classify driver behaviour while driving the car. The system will be embedded in a raspberry pi camera that can detect those behaviours. The camera then monitors the driver and when it detects a behaviour that should not be done during driving, the system will alarm the user.



Objectives:

- The main objective of this project is to reduce vehicle accidents caused by driver distraction on the road, and so save drivers lives.
- This project will reduce the costs of repairing vehicles since accidents rate is also reduced.
- Using high-tech like deep learning in an everyday activity.

Project lifecycle:

Deep learning projects are highly iterative; as we progress through the DL lifecycle, we'll find our self-iterating on a section until reaching a satisfactory level of performance, then proceeding forward to the next task (which may be circling back to an even earlier step). Moreover, a project isn't complete after you ship the first version; you get feedback from real-world interactions and redefine the goals for the next iteration of deployment

Users:

- Drivers.
- Transport companies.
- Driving schools.

Features:

1- The device shall work on a raspberry pi system.
2- The device shall have a camera and a sound alarm connected to the raspberry pi.
3- A CNN deep learning model that is deployed on the system.
4- The CNN model uses transfer learning to get better accuracy and results.
5- The CNN model shall be trained on a dataset the include all the cases the system classifies.
6- The CNN model classifies the driver behaviour on 9 classes (cases):
    - Safe driving.
    - Texting (right hand).
    - Texting (left hand).
    - Talking on the phone (right hand).
    - Talking on the phone (left hand).
    - Reaching behind.
    - Hair and makeup.
    - Talking to passengers.
    - Sleeping.
7- An alarm goes off / on when the system detects a distraction behaviour.
8- The device's price will be in reach of hands.

Steps to complete the project:

1- Pre-processing the datasets and prepare them for training and testing the model.
2- Build the model by using transfer learning with different pre-trained models and choose the best one.
3- Optimize the final model to reach the required accuracy.
4- Build the detection device with raspberry pi, a camera and an alarm sound system.
5- Deploy the model we built to the raspberry pi.
6- Test the device in different cases to insure accuracy.
7- Optimize the device structure and prepare it as a final product.

Datasets:

The datasets will be images collected from different sources and labelled with the class name they represent.

There will be three datasets:

- the training dataset
- the validation dataset
- the testing dataset

Stages of building the model:

1- Model exploration
    - Establish baselines for model performance
    - Start with a Pre-trained model using initial data pipeline
    - Overfit simple model to training data
    - Stay nimble and try many parallel (isolated) ideas during early stages
    - Find SOTA model for your problem domain (if available) and reproduce results, then apply to your dataset as a second baseline
    - Revisit Step 1 and ensure feasibility
    - Revisit Step 2 and ensure data quality is enough.
2- Model refinement
    - Perform model-specific optimizations (ie. hyperparameter tuning)
    - Iteratively debug model as complexity is added
    - Perform error analysis to uncover common failure modes
    - Revisit Step 2 for targeted data collection of observed failures

3- Testing and evaluation
    - Evaluate model on test distribution; understand differences between train and test set distributions (how is “data in the wild” different than what you trained on)
    - Revisit model evaluation metric; ensure that this metric drive desirable downstream user behaviour
    - Write tests for:
        - Input data pipeline
        - Model inference functionality
        - Model inference performance on validation data
        - Explicit scenarios expected in production (model is evaluated on a curated set of observations)

User Stories:

1- As a programmer, I want to collect data, so I need to search the internet for images that represent the classes.
2- As a programmer, I want to clean the data, by removing unclear images.
3- As a programmer, I want to separate data to training, validation, and testing datasets, by writing a code that split them.
4- As a programmer, I want to pre-process the data by resizing, cropping, and augmenting the images.
5- As a programmer, I want the training dataset to be ready for training the model, so I need to pre-process it first.
6- As a programmer, I want to use the validation dataset to validate the model during training, so that I can save the model or stop the training when the validation loss is the minimum.
7- As a programmer, I want to use the testing dataset to test the model accuracy after training and validation by feeding the model the testing images to return the accuracy of the classification.
8- As a programmer I want to try different models and choose the one with the higher accuracy, by doing transfer learning on pre-trained models.
9- As a programmer, I want to choose different pre-trained models and fine tuning them by changing the last fully connected layers by linear layers I define.
10- As a programmer, I want to save the satisfying model and deploy it to the raspberry pi device.
11- As a programmer, I want the device to be capable of running my model efficiently.
12- As a user, I want the device to be portable and easy to use, so the device must be small and light-weighted.
13- As a user, I want the device with a camera connected with it by buying a camera combatable with the raspberry pi.

14- As a user, I want the device to include a smart system that can detect and classify videos and images from the camera.
15- As a user, I want the device to include an alarm sound system to alert me.
16- As a user, I want to be able to switch the device on and off, by include a switch on device.
17- As a user, I want the device to monitor my behaviour while driving.
18- As a user, I want the device to alarm me when I am distracted in any case the device can detect.
19- As a user, I want the alarm to be off when I am in a safe mode or not distracted.

Links:

- Github: https://github.com/hananHA/DriverDistractionDetection
- Trello: https://trello.com/b/7HNLv8eM/driver-distraction-detection