BABEŞ BOLYAI UNIVERSITY, CLUJ NAPOCA, ROMÂNIA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

SMART DRIVING ASSISTANT

– MIRPR report –

Team members
Marc Moldovan, Software Engineering, 258/2
Tudor Maier, Software Engineering, 258/2

2020

Abstract

This paper documents a research in which a new mobile application was developed for Android devices that will work as a smart co-pilot for the drivers. This product introduces real-time object detection for the phone camera on a various set of objects, together with location and weather service integration. The idea of this application is to provide an easy-to-use mobile application with a user-friendly interface that will provide different information about the road.

Contents

Table of contents **1**

1 Introduction **2**
- 1.1 What? Why? How? 2
- 1.2 Paper structure and original contribution(s) 2

2 Scientific problem **4**

3 State of the art **5**

4 Proposed approach **6**
- 4.1 TensorFlow Object Detection API with SSD model 6
- 4.2 Data formatting and collecting 8
- 4.3 Google location service 8
- 4.4 AccuWeather Api 8

5 Application **9**
- 5.1 Methodology 9
- 5.2 Data 9
- 5.3 Results 10
- 5.4 Discussion 14

6 Conclusion and future work **15**

7 References **16**

Chapter 1

Introduction

1.1 What? Why? How?

The auto industry has been growing lately, this automatically sporing the number of drivers. Having this in mind, the outcome has both advantages and disadvantages.

The main disadvantages are pollution and the increase in the number of road accidents caused by careless driving as well as the failure to adjust the speed in different situations. Currently, there are several ways to increase the driving experience, but these are either very expensive or require a high level of technical skills to use them.

Also, the mobile app industry is growing insanely fast and there are more and more smart gadget users. Users prefer using mobile devices applications on behalf of other static devices such as personal computers or web applications because you have access to them at any time.

Applying the reasons mentioned above, creating a mobile application with the purpose of road assistant can increase the safety of the drivers.

In this paper, we want to present an easy-to-use application, which combines several concepts such as: machine learning, location services and third party services.

The application will be able to detect different objects and traffic participants such as: pedestrian crossings, traffic signs, pedestrians; but it will also calculate the speed at which the vehicle is moving and will warn you about adapting the speed to the present weather conditions.

1.2 Paper structure and original contribution(s)

The research presented in this paper advances the theory, design, and implementation of several particular models.

The main contribution of this report is to present a smart algorithm to solve the problem of detecting different objects in traffic as well as traffic participants.

The second contribution of this report consists of building an intuitive, easy-to-use and friendly mobile application. Our aim is to build an assistant that will help drivers identify objects in traffic faster. Based on the results of the algorithm, it is possible to display on the screen the possible dangers as well as what they should be aware of at that time.

The third contribution of the paper is the integration of a third party application that provides weather data with a location service so that the driver can be notified when he has to reduce the speed to adapt to the weather conditions at that time. Moreover, it is desired to notify the driver when it exceeds the speed limit.

The present work contains 10 bibliographical references and is structured in five chapters as follows:

The first chapter is a short introduction in the paper where the reasons behind it are presented and how it is structured.

The second section describes the scientific problem behind the paper and what are the advantages and disadvantages of using machine learning.

The third chapter presents the theories and applications used so far to solve the problem of the work.

The fourth section describes the approach for solving the stated problem.

The fifth chapter presents the developed application and its results.

The sixth section describes the conclusions and future work.

Chapter 2

Scientific problem

Automobiles capable of transporting human people have already been invented for over 100 years ago, therefore, now, people have already gotten accustomed to driving and tend to get distracted due to other external factors, such as other people in the car, phone ringing, listening to music and so on.

Taking into account this matter, the drivers' and pedestrians' safety is at risk and requires another approach, other than the usual laws, which the drivers/pedestrians tend to break them more and more.

Ultimately, having access to easy-to-use Artificial Intelligence tools & algorithms can lead to the development of autonomous cars or system that will prevent the dangers on the road and reduce the risk in traffic.

Chapter 3

State of the art

The need of a smart artificial co-pilot was already noticed by the car manufacturers and all of them started to invest into this direction in the past few years. Each of them tries to find an optimal solution, but it involves different additional costs, for example: attaching new sensors and cameras to the car, collecting data and displaying them in a human-friendly form, attaching a graphical interface on the main console of the car to extend the infotainment system.
BMW has one of the Intelligent Driving Assistant on the market. It offers different features such as camera-based system for detecting vehicles around the car, Person warning for pedestrian detection and notifying the driver of possible dangers. Moreover, they also offer speed limit information depending on the location of the vehicle and recommend new speed limits depending on the weather conditions. In addition, they include many useful features: BMW Night Vision, head-up display, parking assistant and surround view of the car [10].

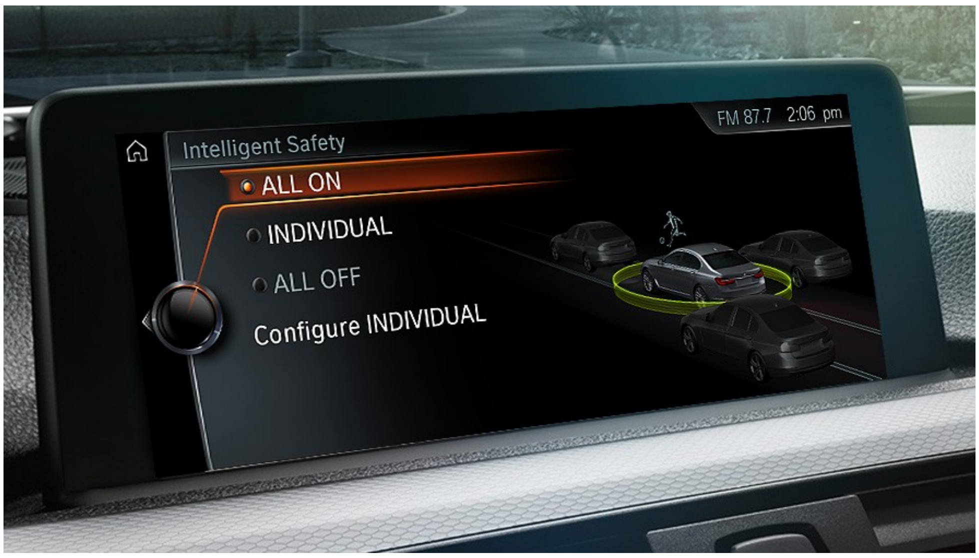

Fig. 3.1. Lane keeping assistant from BMW [10]

Chapter 4

Proposed approach

Driving assistant has been implemented on an Android platform, being one of the most used devices and also offering different useful sensors, such as gps location and camera recording. Moreover, we can make use of the size of this Android device to make it mobile and easy to use and mounted on a vehicle. The Android application aims to detect the different traffic signs (pedestrian crossing sign, traffic light), road markings (pedestrian crossing), as well as pedestrians on the sidewalk or pedestrian crossing.

In order to detect certain signs, objects, markings or traffic participants, an Object detection algorithm is required. So this was our main challenge.

4.1 TensorFlow Object Detection API with SSD model

TensorFlow Object Detection API is “an open source framework built on top of TensorFlow” that aims to make it easy to “construct, train and deploy object detection models” [1]. Along with the API they provide a few pre-trained models such as ssd_mobilenetv2 and faster r-cnn2.

MobilenetV2 is neural network architecture published by Google that was optimized for mobile devices. Mobilenet is a pre-trained model which was trained using COCO Dataset (Common Objects in Context) which consists of over 300 thousand images and over 200 thousand labeled classes. It contains 91 object types such as “person” , “traffic light” , “car” [2].
With the new version of the Mobilenet model, Google provided a new architecture which contains two types of blocks: residual and for downsizing. Also, there are three layers for both types of blocks [3]. Residual blocks connect the beginning and end of a convolutional block with a skip connection. By adding these two states the network has the opportunity of accessing earlier activations that weren’t modified in the convolutional block. This approach turned out to be essential in order to build networks of great depth. MobileNetV2 follows a narrow->wide->narrow approach. The first step widens the network using a 1x1 convolution because the following 3x3 depthwise convolution already greatly reduces the number of parameters. Afterwards another 1x1 convolution squeezes the network in order to match the initial number of channels [4].

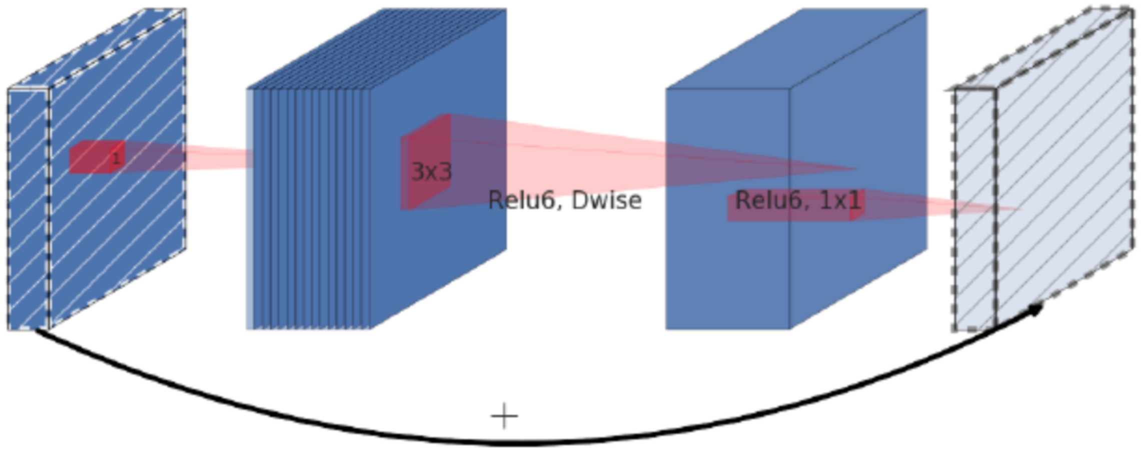

Fig. 4.1 Residual block [4]

Faster R-CNN is one of the most famous detection architecture that uses convolution neural networks like YOLO (You Only Look Once) or SSD (Single Shot Detector) [5]. This model is trained independently with weights from a network trained for an ImageNet classification task. ImageNet is an image database with over 14 millions images and 21841 synset indexed with classes like “person” [6]. Faster R-CNN is composed of three parts.

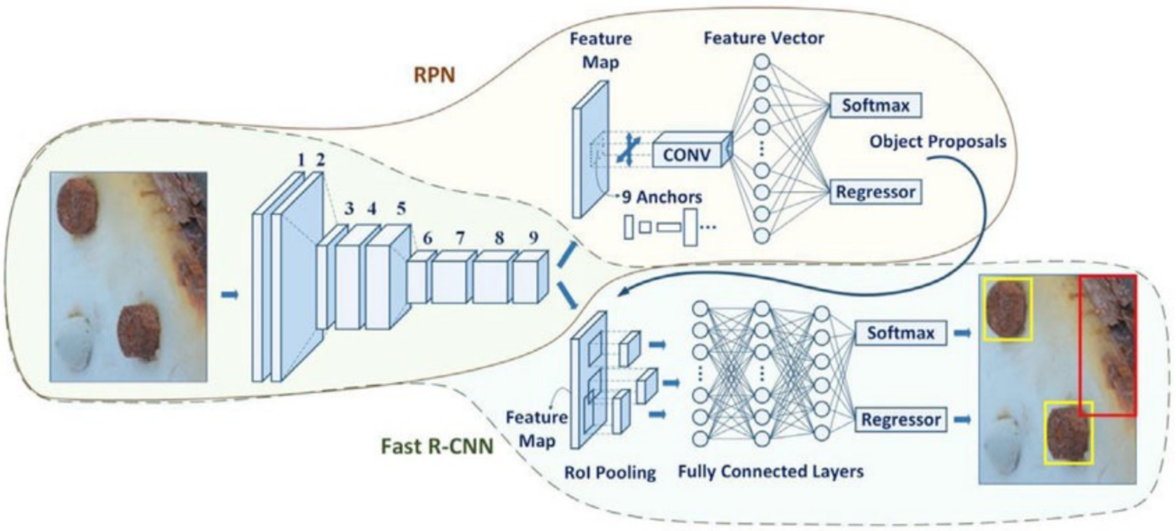

Fig 4.2. Faster R-CNN architecture [5].

First part is the convolutional layer. Here the filters are trained to extract features from image. These filters are filtering our input image and are extracting just a feature map

of the image by eliminating pixels with low values [5].
The second part is the Region Proposal Network (RPN) which is a small neural network sliding on the last feature map of the convolution layers and predict whether there is an object or not and also predict the bounding box of those objects [5].
The last part consists of Classes and Bounding Boxes is defined by taking the input from the RPN and predict object class (classification)

4.2 Data formatting and collecting

The first thing that we had to do in order to train a custom model is to collect data. So we picked a set of images from google maps street view. After selecting different “driving situations” .

Secondly, to improve the learning speed we had we need to resize our images before annotating them. at a resolution of 800 x 800. We have done this with the help of PhotoShop scripts. After the resize, we split the images into two directories: “train” and “test” . We picked a ration between these of 20% - 80% .

Lastly, the annotation phase was done using an open LabelImg tool. This tool, will generate “xml” files with the position of desired object. This tool saves the “xml” files in a format supported by both ImageNet and SSD [7].

4.3 Google location service

One of the most important features of the mobile devices is the location awareness. The location API is available through the Google Play Services and enabling it in the application will provide location tracking, geofencing and activity recognition.
Using the location service we can determine the speed current speed. We can do this by storing the previous location and always compare it with the current one. We used the following formula:

$$s = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \;/\; (t_1 - t_2)$$

s - speed
x1, y1 - current longitude and latitude
x2, y2 - last longitude and latitude
t1, t2 - current time and last location registration time

4.4 AccuWeather Api

AccuWeather is a company that provides weather forecasting services worldwide. It also has an open API that we can consume [9]. Combined with Google Location Services we will get our current location weather data.

Chapter 5

Application

5.1 Methodology

In order to re-train a pre-trained model, images taken from Google Maps with Street View functionality were used in order to obtain real images and a slightly various dataset.
After gathering around 30 images, the next step was to process them in order to achieve better performance time for the training phase, therefore a batch resize was executed with Photoshop in order to reduce the size of the images to around 800px x 800px.
Furthermore, to train the object detection model, the images needed to be labeled in order to detect a specific set of classes. For this phase, an open-source tool was used, named LabelImg, which can be found on GitHub and is easy to use for labeling the objects within a dataset.
The labeled images were split into 2 groups, one for training phase and one for testing phase, the recommended ratio is said to be somewhere around 80% training images and 20% testing images. An important mention here, is the fact that the testing images must be different from the training images in order to receive correct results of the model's efficiency.

5.2 Data

The data used was simply taken from Google Maps using the Street View functionality. The images were taken from various angles, zoom level and distance in order to efficiently train the model.
The images are real images taken from the real world, therefore the obtained model should be more accurate.

5.3 Results

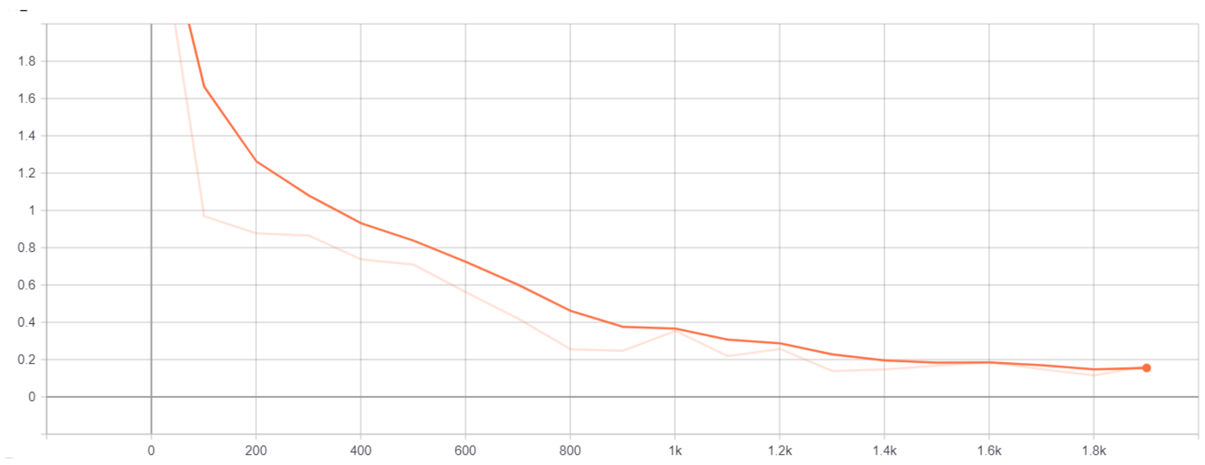

Figure 5.1 (Faster R-CNN Inception V2 2000 steps training loss graph)

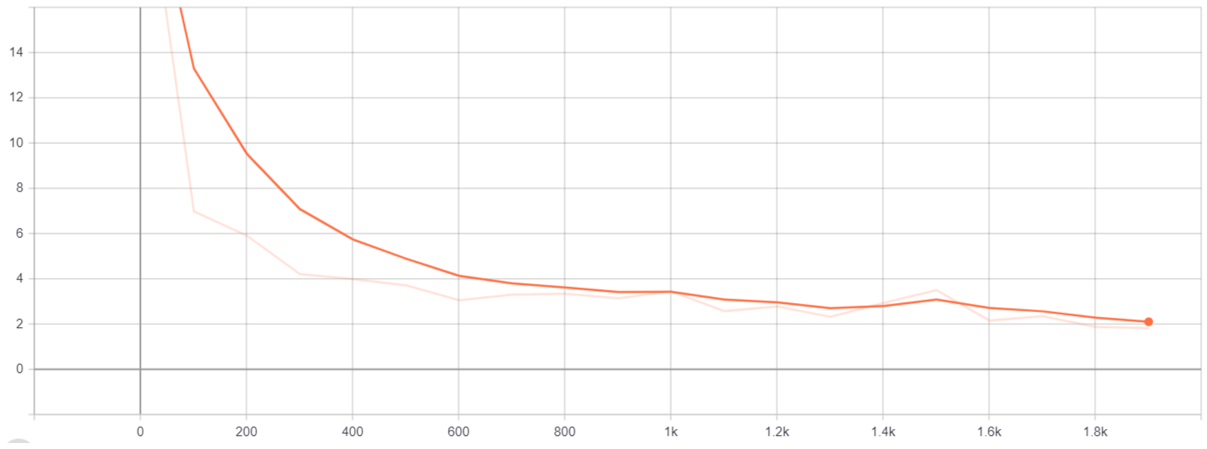

Figure 5.2 (SSD Mobilenet V2 2000 steps training loss graph)

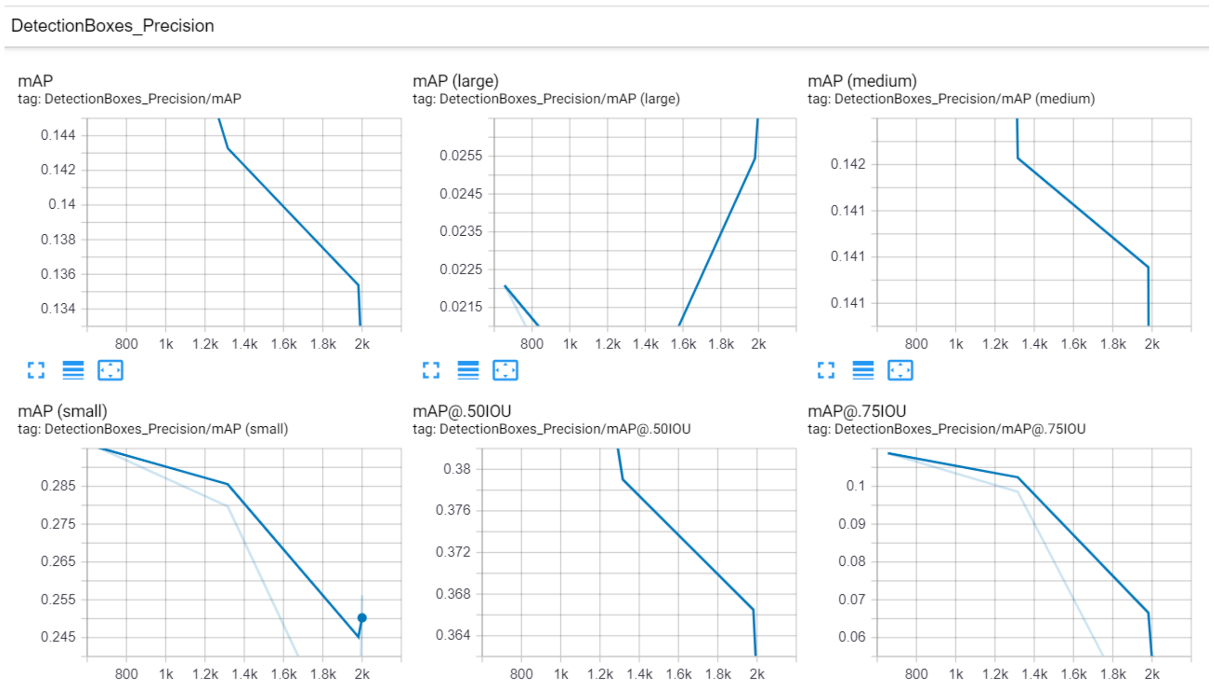

Figure 5.3 (Faster R-CNN Inception V2 2000 steps precision/mAP graph)

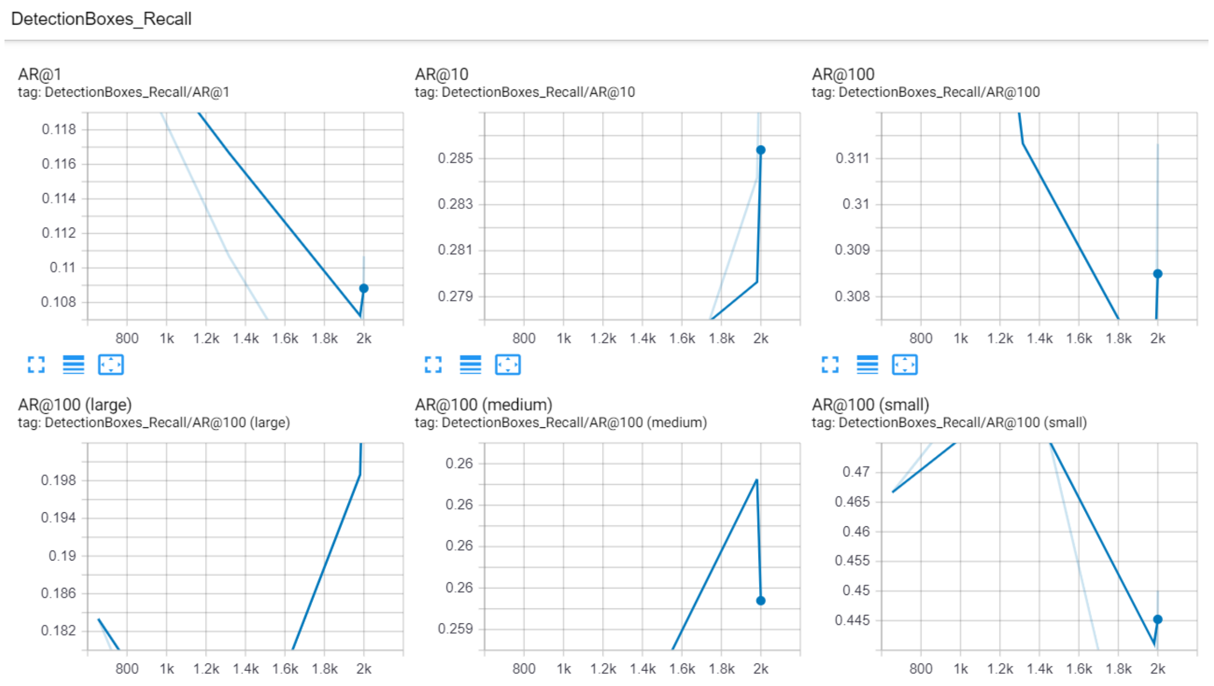

Figure 5.4 (Faster R-CNN Inception V2 2000 steps recall graph)

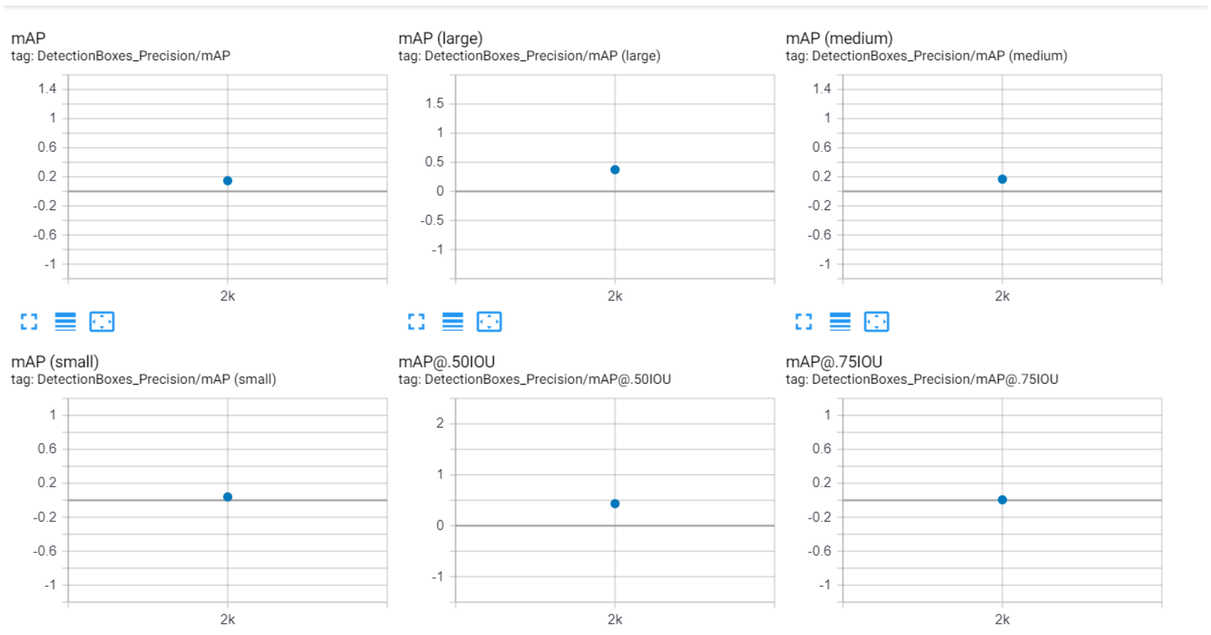

Figure 5.5 (SSD Mobilenet V2 2000 steps precision/mAP graph)

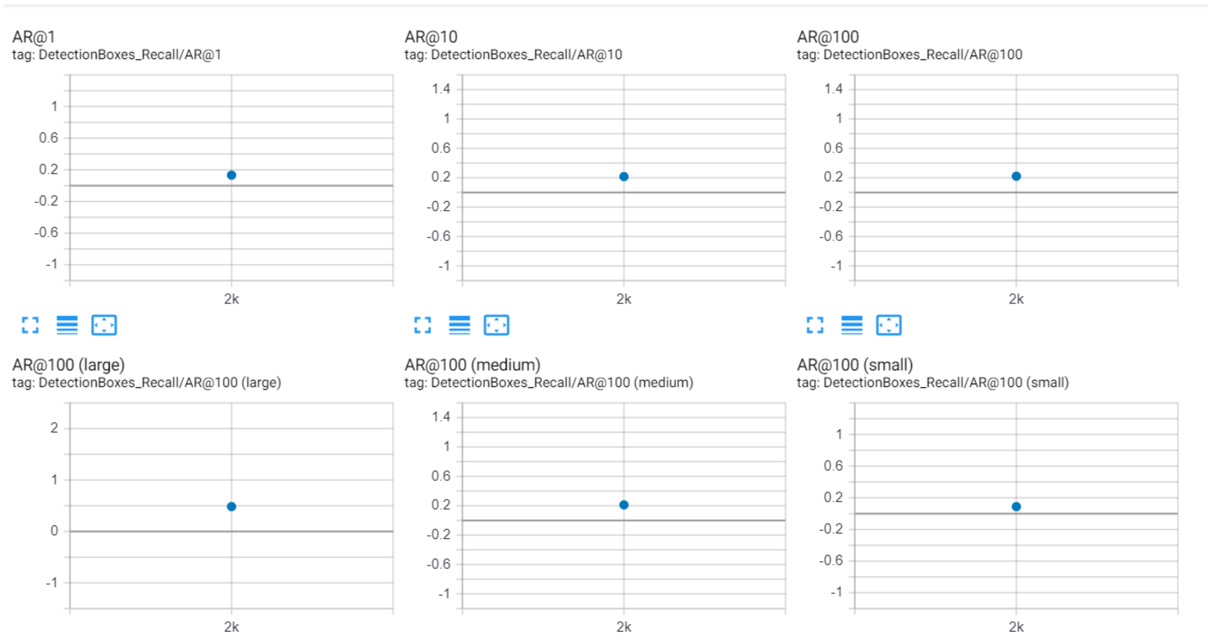

Figure 5.6 (SSD MobilNet V2 2000 steps recall graph)

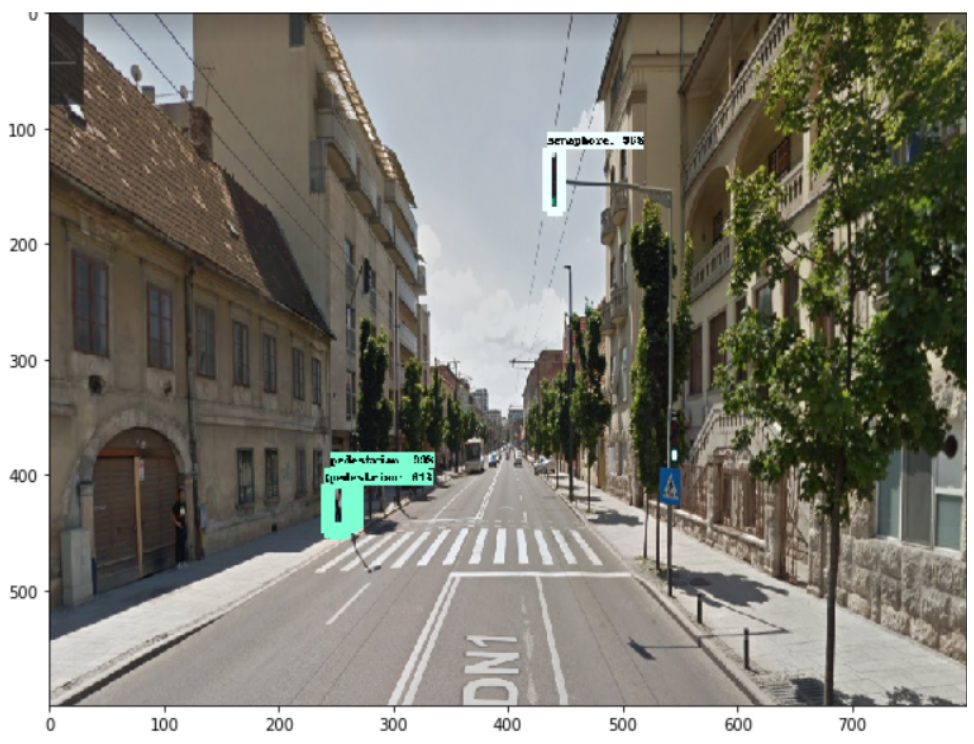

Figure 5.7 Faster R-CNN test image 1

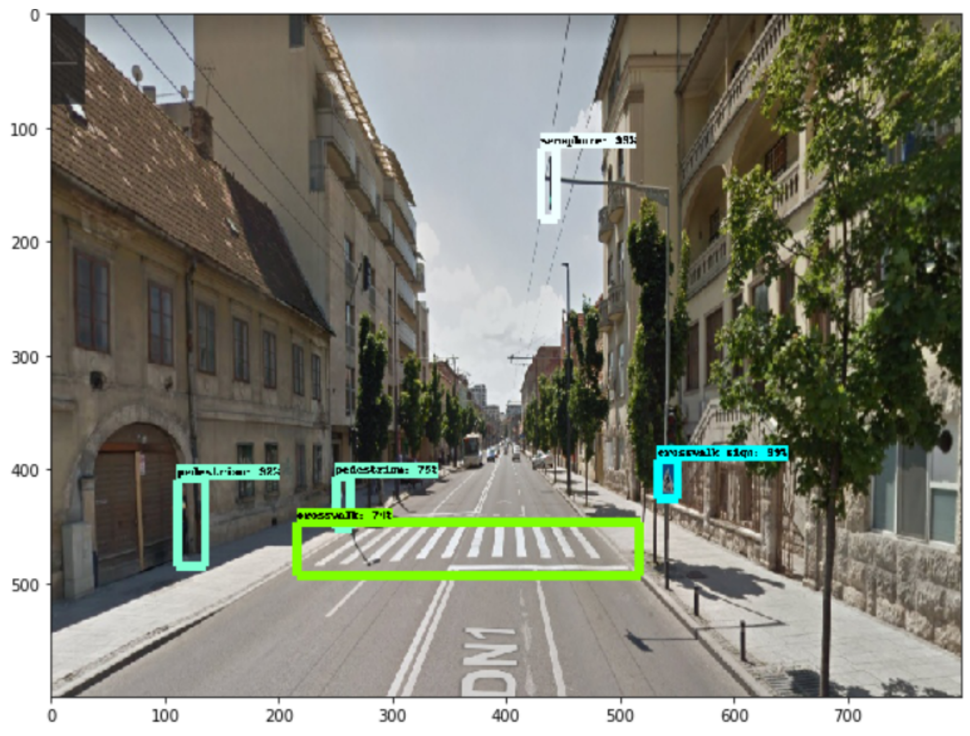

Figure 5.8 SSD MobileNet test image 1

5.4 Discussion

The graphs displayed in the previous subchapter, where 2 pre-trained models (SSD Mobilenet and Faster R-CNN) were used for transfer learning, depict the average precision, average recall and loss function. Taking into account these data, the Faster R-CNN has an average precision of 13,4% (Figure 5.3) while the SSD Mobilenet has a slightly higher average of around 14% (Figure 5.5). These values are very low compared to the precision of the pre-trained model of 90% for Faster R-CNN and 61% for SSD Mobilenet mainly because of the very small number of training steps. In order to achieve comparable results to the results of the pre-trained models the number of training steps must be increased.

There have been made benchmarks on pre-trained models and one of them shows that Faster R-CNN Inception V2 obtains a higher mAP than the SSD Mobilenet, the latter one is the fastest and lightest model in terms of memory consumption, making it an optimal choice for deployment in mobile and embedded devices, however Faster R-CNN Inception V2 is also the lightest model between the R-CNN architecture.[8]

model	MB	Total exec (ms)	CPU(ms)	mAP
Faster R-CNN Resnet 50	*5256.454615*	*104.0363553*	*28.10240132*	*91.52*
Faster R-CNN Resnet 101	*6134.705805*	*123.2729175*	*32.9357732*	*95.08*
Faster R-CNN Inception V2	2175.206857	58.53338971	19.76525	90.62
Faster R-CNN Inception Resnet V2	*18250.446008*	*442.2206796*	*76062*	*95.77*
R-FCN Resnet 101	*3509.75153*	*85.45207971*	*33.04886232*	*95.15*
SSD Mobilenet	94.696119	15.14525	11.12398214	61.64
SSD Inception V2	*284.512918*	*23.74428378*	*14.35087838*	*66.10*

During the re-trainings of the pre-trained models SSD Mobilenet V2 and Faster R-CNN Inception V2 there's been a major difference between them, SSD is trained 3 times faster than the Faster R-CNN making it easier for developers to re-train the model faster and tweek the input parameters in order to achieve the best results in terms of accuracy.

Chapter 6

Conclusion and future work

In conclusion, a mobile application using Artificial Intelligence can reduce the risk of potential danger events in traffic by informing the driver ahead of time. A mobile application is easy to use and can easily be used in any vehicle.
Moreover, as time passes by, AI algorithms and models increase their efficiency, making these kind of applications more trustful and efficient.
A possible future improvement is to add more traffic signs to the model in order for the system to identify all possible route dangers.
Taking into account the fact that more traffic signs can be recognized, an algorithm can be implemented, in order to send more accurate danger signals taking into consideration more factors, such as weather conditions, driver's current state of mind, car's real-time statistics (speed, car defects, and so on).

Chapter 7

References

[1] The TensorFlow Authors (2019) TensorFlow Object Detection API (readme). URI: https://github.com/TensorFlow/models/tree/master/research/object_detection.

[2] Common Object in Context (2019) documentation. URI: http://cocodataset.org/#home

[3] Review: MobileNetV2 — Light Weight Model (Image Classification), Sik-Ho Tsang, May 19, 2019,
URI:https://towardsdatascience.com/review-mobilenetv2-light-weight-model-image-classification-8febb490e61c

[4] MobileNetV2: Inverted Residuals and Linear Bottlenecks, Paul-Louise Pröve, URI: https://towardsdatascience.com/mobilenetv2-inverted-residuals-and-linear-bottlenecks-8a4362f4ffd5

[5] Faster RCNN Object detection, Achraf KHAZRI, Apr 9, 2019, URI:
https://towardsdatascience.com/faster-rcnn-object-detection-f865e5ed7fc4

[6] ImageNet (2019) Documentation, URI: http://www.image-net.org/

[7] LabelImg (readme). URI: https://github.com/tzutalin/labelImg

[8] Traffic Sign Detection, https://github.com/aarcosg/traffic-sign-detection

[9] AccuWeather, https://en.wikipedia.org/wiki/AccuWeather
[10] BMW Intelligent Driving, https://www.bmw.ca/en/topics/experience/connected-drive/bmw-connecteddrive-driver-assistance.html