

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

# Traffic Signs Recognition

– ITSG report –

## **Team members**

Brici Robert

Hojda Catalin

Oncioiu Costin

Onisor Ioana

Software Engineering, 258

### **Abstract**

According to statistics, Romania ranks highest in the whole European Union when it comes to the number of fatal car crashes. [CE 2018] Given this issue, we believe that intelligent software system designed for the automotive industry have the potential to provide significant improvements when it comes to this tragic pattern that our country is following. One main focus point when it comes to the AI development in this industry is the recognition of the traffic signs, which are the key factor by which traffic is regulated. They control its flow, provide information regarding how drivers should behave, inform a driver about the directions and distances along with guides to a destination, or warn him/her about specific dangerous places. Given the increasing a number of drivers, a need to make a road more safe arises.

# Contents

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Introduction</b>                                    | <b>1</b>  |
| 1.1      | What? Why? How? . . . . .                              | 1         |
| 1.2      | Paper structure and original contribution(s) . . . . . | 2         |
| <b>2</b> | <b>Scientific Problem</b>                              | <b>3</b>  |
| 2.1      | Main functionalities of the application . . . . .      | 3         |
| <b>3</b> | <b>State of art/Related work</b>                       | <b>6</b>  |
| <b>4</b> | <b>Proposed approach</b>                               | <b>8</b>  |
| <b>5</b> | <b>Application (numerical validation)</b>              | <b>11</b> |
| 5.1      | Methodology . . . . .                                  | 11        |
| 5.2      | Data . . . . .   | 13        |
| 5.3      | Results . . . . .                                      | 14        |
| 5.4      | Discussion . . . . .                                   | 15        |
| <b>6</b> | <b>Conclusion and future work</b>                      | <b>17</b> |

# List of Algorithms

|   |   |    |
|---|---|----|
| 1 | <i>BuildingtheTensorFlowGraph</i> . . . . .             | 9  |
| 2 | Create a session and run the graph we created . . . . . | 9  |
| 3 | Using the Model . . . . .                               | 10 |

# Chapter 1

## Introduction

### 1.1 What? Why? How?

One main focus point when it comes to the AI development in this industry is the recognition of the traffic signs, which are the key factor by which traffic is regulated. They control its flow, provide information regarding how drivers should behave, inform a driver about the directions and distances along with guides to a destination, or warn him/her about specific dangerous places. Given the increasing a number of drivers, a need to make a road more safe arises. An unnoticed sign board can cause fatal accidents. The reasons why they go unnoticed varies from actual intention to Wavering concentration, tiredness or sleep deprivation or even factors non-imputable to the driver, like poor street illumination, an influence of the exterior environment and weather conditions. All these become threats, especially when driving alone. Given the information presented prior, we put forward an idea of developing an AI-based application that is built on top of algorithms that âlearnâ to recognize traffic signs and other threats that might appear when driving. The reason for that is to provide warning for careless drivers. Intelligent Transport Systems have a great potential to save time, money and ultimately lives, and to improve our overall driving experience, making it more safe. Camera-based traffic-sign recognition systems identify signs in real-time by processing the videos/ pictures that are captured through a camera. They help the driver by providing warnings, commands and, if further developed, sometimes by taking control of the vehicle itself. A traffic sign recognition system based on vision will have 2 main modules the detection module and the recognition module. The detection phase consists of pre-processing the image, along with enhancing and segmenting it according to its original properties such as color or shape, as described by law. The output is a segmented image containing potential regions which are considered candidates to be labeled as road signs. The efficiency and speed of the detection are important factors which play a strong role in the whole process. In the recognition

---

step, the candidates are tested against a pattern to decide whether it is an actual road sign or not. After this analysis the classification is made. These patterns are chosen so as to emphasize the differences between each sign.

## **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 an intelligent algorithm for solving the problem of traffic sign recognition.

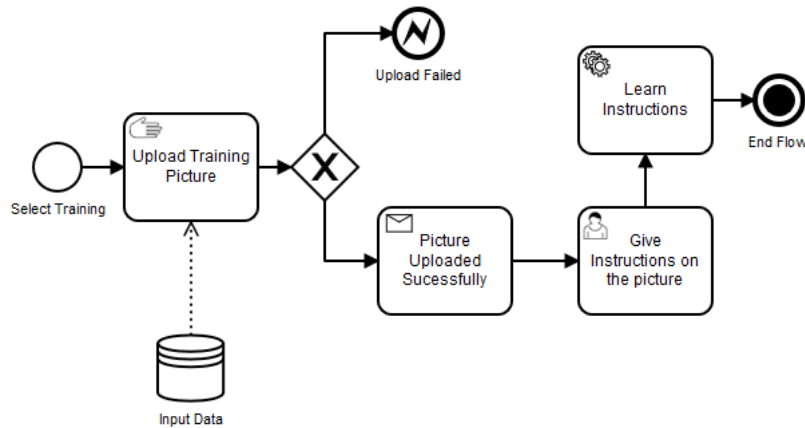
The second contribution of this report consists of building an intuitive, easy-to-use and user friendly software application. Our aim is to use an algorithm that will help identifying if a traffic sign is present in a picture and also that it is.

## Chapter 2

# Scientific Problem

### 2.1 Main functionalities of the application

**Training Mechanism** The training functionality serves as an information provider. By analyzing a large number of pictures/ videos, the application is able to learn through its algorithm all the traffic signs and their meaning and establish a pattern, which the recognition feature will be able to use in order to provide warnings in real time use. The workflow of this feature is presented in Figure 1.

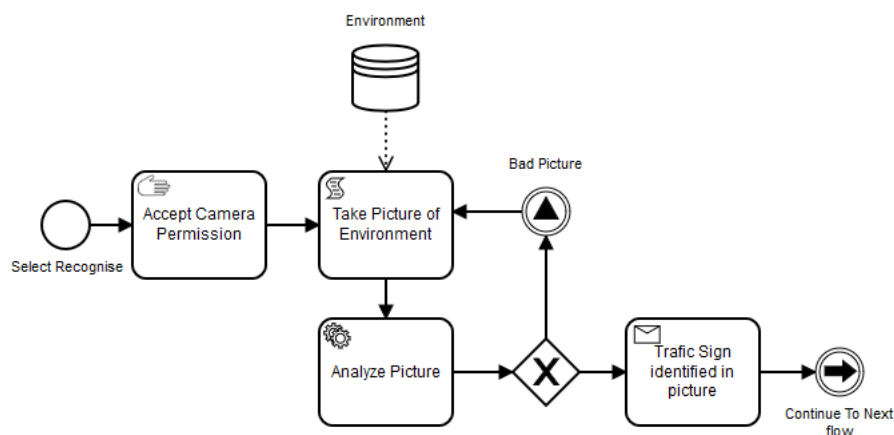


The steps of the process involved in using this feature are the following:

1. Selecting the feature: from the main menu, the user has to click the button dedicated to entering this section of the application
2. Image uploading: the user should upload a file from its own device; the process can only continue if the upload finishes successfully

- 
3. Image upload success: the user is notified that the uploading of the picture to the application s training mechanism occurred successfully and the application is able to further process it
  4. Instructions providing: the user is asked to give details/ tags that the application will further associate to other pictures, by comparison
  5. Learning: the application stores the instructions and is now able to accept new requests

Recognition Mechanism Based on the information provided by uploading a large number of files to the training mechanism, the application should be able to apply the stored information and use it to identify similarities between images taken in real time and legacy data. The main goal is to be able to deliver warnings based on the conflicts identified between the driver s behavior and the laws behind each sign. The first step in achieving this is to make the application identify the signs correctly though. The first park of the recognition flow is taking a picture that the application can process and take information from. This process is described in Figure 2.



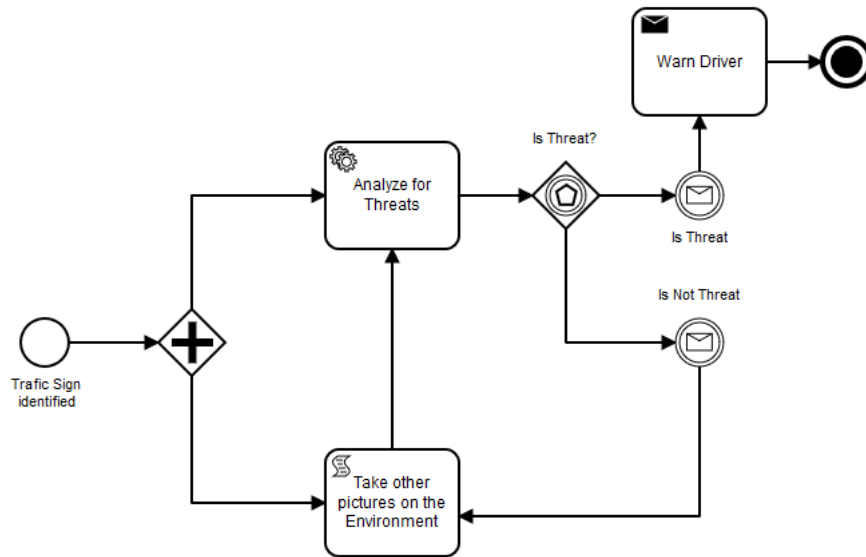
The steps of the process involved in going through the first flow of the feature are the following:

1. Accepting the permissions: the users should let the application user the camera for taking the photos and they have the option to refuse, but the process can t continue then, so it is not even marked in the diagram
  2. The taking of the picture: this is a process that should happen automatically or manually and the picture is then submitted for review
  3. Picture review: the application reviews the picture if it has any segments that can be marked as candidates for being traffic signs and if there are no such segments or the picture has a bad quality, then it is rejected
-



- 
4. Candidate review: each candidates is tested against the the training data and the application determines if it is a traffic sign and if it is, the process continues with phase 2

The second part of the flow concerns identifying if the reviewed candidates makes ending warning signs a necessity and if yes, doing so, as described in Figure 3.



The steps of the process involved in going through final flow are the following:

1. Threat Analysis: check whether the identified issue/ sign is work sending a signal; if it is not, it processes the next picture
  2. Warning: if the applications identifies a threat, it warns the driver
-

## Chapter 3

# State of art/Related work

Traffic assistants are one of the main focus points when it comes to artificial intelligence research and development. Such mechanisms have also been applied to other fields such as Military Decision Making Cycle, Biomedical Engineering, Power Electronic, e-Health, Cybersecurity.

From all the research papers done on this issue, we mention the following:

### 1. **Research Article:** Intelligent Driving System Using Artificial Intelligence

**Author:** Chetan Bulla

**Presented At:** INTERNATIONAL JOURNAL OF RESEARCH IN ELECTRONICS AND COMPUTER ENGINEERING (IJRECE), June 2019

**Algorithms and Modules:**

- Modules: Raspberry PI 3 model B+, Arduino UNO MEGA 3, PI Camera, System
- Algorithm: Cascade Classifier (HaaropenCV): This algorithm mentioned above deals with adjacent rectangular areas at a certain location in a detection window. It adds up the pixel intensities in each area and calculates the difference between these sums. This difference is then used to differentiate subsections of an image.

**Results:** The prediction on the testing samples returns an accuracy of 0.85

**Link:** [ResearchGate](#).

### 2. **Research Article:** Traffic signs recognition for driving assistance

**Authors:** Yatham Sai Sangram Reddy, Devareddy Karthik, Nikunj Rana, Jasmine Pemeena Priyadarsini, G K Rajini, Shaik Naseera

**Algorithms and Modules:**

- 
- Algorithm: HAAR Cascade Training

**Results:** For the traffic sign boards with single contour, the nearest neighbour is found and the output is obtained from the response returned by the classifier. For the traffic sign boards with multiple contours, the respective nearest neighbours are found and the responses returned by the classifier are sorted based on their x-positions.

**Link:** [ResearchGate](#).

3. **Research Article:** A Review of Intelligent Driving Style Analysis Systems and Related Artificial Intelligence Algorithms

**Author:** Herman Myburgh

**Presented At:** Department of Electrical, Electronic and Computer Engineering, University of Pretoria, South Africa, December 2015

**Algorithms:** Artificial Neural Networks, Fast Fourier Transform, State Machines, Hidden Markov Models, Bayesian Networks, k-Nearest Neighbour Classifiers, Fuzzy Logic

**Goal:** To have a risk score for each driver according to their behaviour in traffic. the average score would be around 75.

**Link:** [ResearchGate](#).

4. **Research Article:** Driving Situation-based Real-Time Interaction with Intelligent Driving Assistance Agent

**Author:** Young-Hoon Nho

**Presented At:** Conference: Robot and Human Interactive Communication (RO-MAN), August 2015

**Algorithm:** hidden Markov models (HMMs)

**Goal:** To analyze driving patterns and identify driving situations based on how the functionalities of the car are used, as such: Speed Bump, Corner, Downhill, Crowded Area, Parking Space, Straight, Uphill

**Results:** the system correctly recognized 408 of 430 (or 94.9 percent) driving situations

**Link:** [ResearchGate](#).

---

## Chapter 4

# Proposed approach

The tool of choice for implementing the system described in the chapters before is TensorFlow, which is essentially a framework for building Deep Learning Neural Networks. TensorFlow can train and run deep neural networks for handwritten digit classification, image recognition, word embeddings, recurrent neural networks, sequence-to-sequence models for machine translation, natural language processing, and PDE (partial differential equation) based simulations. Best of all, TensorFlow supports production prediction at scale, with the same models used for training.

TensorFlow allows developers to create dataflow graphs, which are structures that describe how data moves through a graph, or a series of processing nodes. Each node in the graph represents a mathematical operation, and each connection or edge between nodes is a multidimensional data array, or tensor.

TensorFlow has a built-in API for:

- Linear regression: `tf.estimator.LinearRegressor`
- Classification: `tf.estimator.LinearClassifier`
- Deep learning classification: `tf.estimator.DNNClassifier` (which is of main concern for us)
- Deep learning wide and deep: `tf.estimator.DNNLinearCombinedClassifier`
- Booster tree regression: `tf.estimator.BoostedTreesRegressor`
- Boosted tree classification: `tf.estimator.BoostedTreesClassifier`

In order to set up the data that TensorFlow will use in image recognition, we have 2 options:

1. Load data directly in memory: usable for small data sets, as in our project

- 
2. Load data through TensorFlow pipeline: usable for big data sets, as a live version of this application would use. The pipeline will load the data in batch, or small chunk. Each batch will be pushed to the pipeline and be ready for the training. Building a pipeline is an excellent solution because it allows you to use parallel computing. It means TensorFlow will train the model across multiple CPUs.

**Steps undertaken to create a functional TensorFlow sample:**

1. Create a Minimum Viable computational Model, represented as a dataflow graph, as follows

---

**Algorithm 1** *Building the TensorFlow Graph*

---

**BEGIN**

**Require:**  $graph \leftarrow tf.Graph()$

**Ensure:**  $graph.as_default()$  {Placeholders for inputs and labels.}

@  $images_{ph} = tf.placeholder(tf.float32, [None, 32, 32, 3])$

@  $labels_{ph} = tf.placeholder(tf.int32, [None])$

{Flatten input from: [None, height, width, channels]}

@  $images_{flat} = tf.contrib.layers.flatten(images_{ph})$

{Fully connected layer Generates logits of size [None, SignTypes]}

@  $logits = tf.contrib.layers.fully_connected(images_{flat}, SignTypes, tf.nn.relu)$

@  $predicted_{labels} = tf.argmax(logits, 1)$

{Define the loss function}

@  $loss = tf.reduce_mean(tf.nn.sparse_softmax_cross_entropy_with_logits(logits = logits, labels = labels_{ph}))$

{ Create training op}

@  $train = tf.train.AdamOptimizer(learning\_rate = 0.001).minimize(loss)$

**END**

---

2. Training the graph

---

**Algorithm 2** Create a session and run the graph we created

---

**BEGIN**

**Require:**  $session \leftarrow tf.Session(graph \leftarrow graph)$

**Ensure:**  $= session.run([init])$

**for** i=1 TO PhotoCount **do**

$lossValue \leftarrow session.run([train, loss], feed\_dict = images_{ph} : images_a, labels_{ph} : labels_a) RandomlySelectPart$

**if** i divided by 10  $\leftarrow 0$  **then**

    print lossValue

**end if**

**end for**

**END**

---

Note! We use a loss function to determine how far the predicted values deviate from the actual values in the training data. We change the model weights to make the loss minimum, and that is what training is all about.

---

- 
3. Usage of the trained model on other data Now we obtained a trained model stored in the Session object. We can now call `session.run()` on other data. The predicted labels operator returns the output of the `argmax()` function. The goal now is to run it on a set of pictures and compare the label expected by our recognition mechanism to the real one and print both values.

---

**Algorithm 3** Using the Model

---

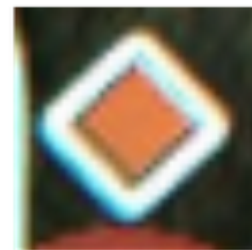
```
BEGIN {Pick some random images}
sampleIndexes  $\leftarrow$  random.sample(range(len(images32)), NoOfNewImages)
sampleImages  $\leftarrow$  [images32[i] for i in sampleIndexes]
sampleLabels  $\leftarrow$  [labels[i] for i in sampleIndexes]
{Run the predictedLabels operator and print results}
predicted  $\leftarrow$  session.run(predictedLabels, sampleImages)
print sampleLabels
print predictedLabels
END
```

---

The result of the above algorithm is a display of the expected label and the presumed one, and if the values match, they are displayed in green and if not, in red.



Truth: 39  
Prediction: 39



Truth: 61  
Prediction: 7

4. Evaluation of the application's accuracy The outcome of this step is to evaluate the precision of the algorithm by comparing the number of labels that match to the ones that do not. Given the fact that, at this point the application has not undergone training, the average scores here tend to be low.
-

## Chapter 5

# Application (numerical validation)

### 5.1 Methodology

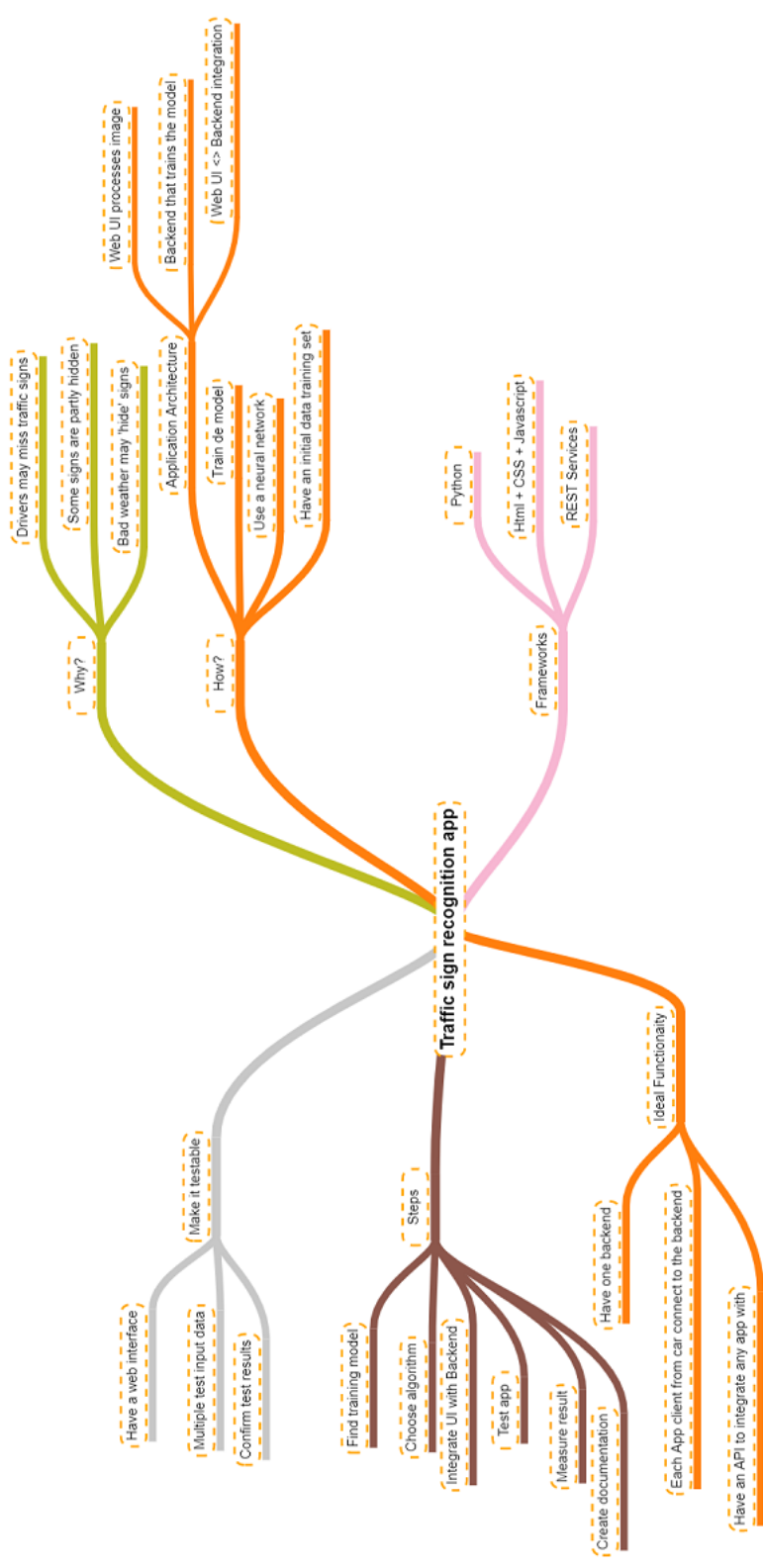
- What are criteria you are using to evaluate your method?

Evaluation of the application s accuracy is done by comparing the number of labels that match to the ones that do not, as described in the previous chapter.

- What specific hypotheses does your experiment test?

The project was organized according to the mind map diagram, which focuses on the following items

1. Why? - the premise of this research is the assumption that some traffic signs are easy to be missed for various reasons.
2. How? - the need for a way to avoid such problems can be fulfilled with an application based on a neural network which is trained with real examples
3. Steps - the projects started with choosing a framework and suitable algorithm whihs we trained and then integrating it with an application which is later tested in the real world.





- 
- What are the dependent and independent variables?

| Independent variables                        | Dependent variables |
|--|---------------------|
| The Core-system                              | -                   |
| The environment in which the system operates | Analyzed photos     |
| The input data (training photos)             | Loss value          |
| Label List                                   | Truth variables     |
| AI Algorithm                                 | AI Analysis results |

- What is the training/test data that was used, and why is it realistic or interesting?

The training/test data that was used is a list of photos that contain traffic signs. The photos are taken from the real world so they serve as a good example on how such an application would run if it was integrated in a driving assistant system.

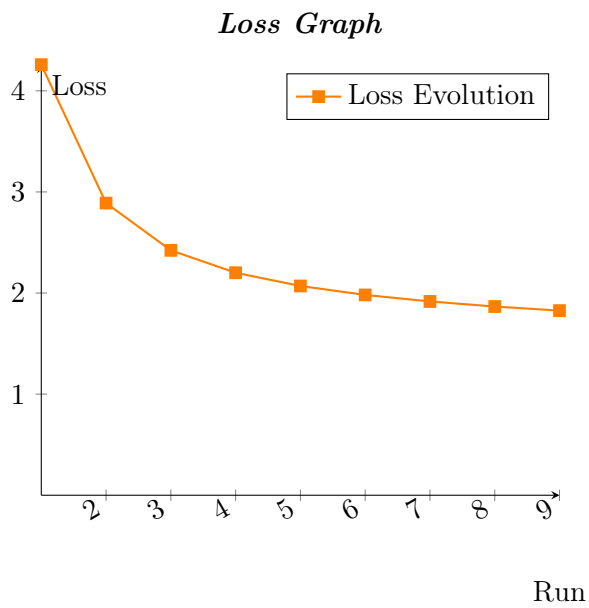
## 5.2 Data

The training data consisted of a set of photos containing traffic signs which was ran through the algorithm many times, in order to iteratively train the model to minimize the loss function.

The loss function has an important job in that it must faithfully distill all aspects of the model down into a single number in such a way that improvements in that number are a sign of a better model.

The results calculated by the loss function through each iteration are described in the graph below (the first 9 iterations) and show that the value (which was printed after each iteration, for reference) keeps decreasing after each one of them.

---







### 5.3 Results

The Accuracy we got when running the algorithm for different sets of data was around 0.6. While analyzing the falkyness of the system, we noticed certain patterns on what the system is able or not to identify . We were able to narrow the issues that the application has to the following types

- Traffic signs confusion makes the application mistaking one sign for another
- The application being unable to identify different patterns of similar signs
- The application crashes in certain situations - mainly when the input photo contained more that one traffic sign

Some examples of confusions are the following

| Error Sample  | Mistaken for  | Details  |
|---|---|--|
|  <p>Truth: 61<br/>Prediction: 7</p>  |  | <p>The mistake might have arisen due to the fact that this traffic sign is unique in both shape and color display.</p> |
|  <p>Truth: 52<br/>Prediction: 37</p> |  | <p>This mistake might arise from the fact that the residential area sign has too many elements inside it.</p>          |

The most common failure in identifying subtle differences was in the case of traffic signs with similar shape and color, but with different meaning, such as those depicting different speed limits. As it can be seen, the cause of this is lack of training, since they are marked as correctly identified with the same label.

|   |  |
|---|--|
|  <p>Truth: 32<br/>Prediction: 32</p> |  <p>Truth: 32<br/>Prediction: 32</p> |
|---|--|

## 5.4 Discussion

- Is your hypothesis supported?

Given the fact that the application fulfills its intended purpose, the hypothesis is supported. Any errors and issues regarding its way of working or any lacking features at his time can be subjected to further improvement or implementation in order for it to fully fulfil its projected usage. It has been proven that such an application can indeed assist a driver.

- What conclusions do the results support about the strengths and weaknesses of your method compared to other methods?

---

The biggest threat for the driver when working with such an assistant is having one traffic sign mistaken for another. In our case study, the results were weaker than the ones got by the professionals mentioned in the state of art.

Given the experience of the team in the field, the fact that we were able to use TensorFlow, a 3rd party application, can be considered a strength, since the algorithm was already implemented and tested and we had to adapt it to our area of concern.

- How can the results be explained in terms of the underlying properties of the algorithm and/or the data.

Many of the errors thrown by the application can be blamed on the lack of proper training. Running the algorithm through more examples can help improving its accuracy.

---

## Chapter 6

# Conclusion and future work

In conclusion, we believe that we were able to demonstrate with our approach, first, that creating an application that is able to recognise traffic signs is possible. This was considered to be the main goal of this project.

During our work we ran TensorFlow with its recognition algorithms on a few sets of pictures containing traffic signs and based on the accuracy of it, managed to identify a set of strengths and weaknesses of this approach.

We believe that developing a fully functional application that is able to deliver real-time will become a focus point of the A.I. development in the future and the motives are the following

- The existence of such an application can provide aid to drivers, which as human beings, tend to easily lose focus sometimes or be absolutely reckless when behind the steering wheel
- The application can either be a stand alone product or be embedded inside a GPS System
- Test data is abundant

When developing our attempt of creating such an application, we believe that our main strongholds are as follows

- Tensor Flow is a solution that was already developed and tested by professionals
- Tensor Flow is better documented than other A.I. tools
- Tensor Flow can be used to a certain degree free of charge

When trying to run the algorithm for a given set of data, the accuracy of our results was lower than the ones obtained by the researchers mentioned at Chapter 3. Many of the problems we faced at

---

this level were, however, caused by the lack of rigorous training. The main threats of a running system of this type that we identified are listed below

- The application tends to mistake some traffic signs for others, particularly when they are similar in shape, but different in color.
- The application tends to crash when there is more than 1 traffic sign inside a given photo, which is often the case in real life
- The application can not prioritize traffic signs, which would be a big problem if we want to also deliver warnings to drivers, as it is the case, for example, in the situation in which we have both traffic lights and priority signs, the traffic light having power over the sign
- Such an application would hardly be able to identify a policeman making specific gestures that can influence the traffic
- It would be hard for the system to identify traffic signs that have more than two components, as it is the case, for example, with the arrows placed below a sign marking a no parking zone

Regarding any future development in this area, we believe that development in this area should stop at the given concept and do not go further. We only want a system that gives plain warnings when a driver encounters a sign and nothing more. The application should not keep notes on whether or not the driver respects the warning whatsoever. We strongly disagree with keeping track of the behaviour of the drivers, as we consider that this is against the laws grating human liberty. Keeping aside the fact that driving conditions might differ from time to time and from place to place, making judging someone by whether or not he or she respects all signs is irrelevant, we believe that such data can be used for malicious interest, such as increasing insurance rates for some drivers, regardless of the fact that ignoring certain signs does not lead to any inconvenience at all for anyone.

---