Embedded System for Real-Time Traffic Monitoring on Smart Cities

Guilherme Lima
INESC-ID, Instituto Superior Técnico, Lisboa, Portugal
Email: guilherme.lima@tecnico.ulisboa.pt.

Abstract—Smart cities depend on various intelligent systems to improve living conditions on the urban areas. Intelligent Transportation Systems are important as they provide different transportation mechanisms which cooperate to optimize routes, reduce traffic congestion, travel time, pollution, waste, or introduce differentiated services such as tolls and surveillance. Often such systems are based on IoT solutions to acquire data from the environment. This work is focused on road transportation, and the ability of classifying a passing vehicle, which plays a key role in traffic monitoring applications and tolling systems. It introduces a novel classification system based on measurements of the light from a vehicle’s headlights. The proposed system relies on a simple photo sensor to classify a passing vehicle. The sensor captures and measures the light emitted by a vehicle’s headlights and then extracts its features and classify them as a signature of the perceived light variation. A neural network is trained to perform the classification process. The proposed method was implemented on a SoC and evaluated on a real-life scenario, including two most common types of vehicles: cars and buses. Preliminary results show that this system was able to correctly classify vehicles in real-time with an accuracy of nearly 80%. Such results make this methodology a promising technique to be further studied and extended to other classes of vehicles, environments or even applications.

Index Terms—Smart-cities, Intelligent Transportation Systems, IoT, Embedded Systems, Low-Power, Real-Time, Classification, Neural Networks, FPGA.

I. INTRODUCTION

Most of the major cities around the world are creating and deploying smart city strategies [23]. Some of these strategies foster the development of a global Internet of Things (IoT) for shared digital services, common IoT architectures, and the Interoperability Mechanisms necessary to support the concept of Open and Agile Smart Cities (OASC) [19], [14]. IoT enabled cities are expected to become smarter and to offer new digital services, ranging from smart grids to smart mobility.

The present work is focused on an IoT system for Intelligent Transportation Systems, more specifically on classification of traffic in real-time. Vehicle classification is of great importance for surveillance, road planning and maintenance, calculate toll charges [2], traffic congestion and collision monitoring [13]. [24] presents a survey on data-driven Intelligent Transport Systems (ITS) available.

Fig. 1 illustrates a typical urban traffic scenario. In this figure it is possible to identify common urban furniture such as traffic lights, speed limit indicators and excess-speed radars. With the advent, and proliferation of IoT devices, it is possible to obtain data about drivers habits, and provide information to drivers to improve their commute routes and minimize hazards and delays.

One of the main challenges in mass deploying novel IoT solutions on a large city concerns: connection over large distances, power restrictions, pollution, long periods without maintenance, and road works required for installation. The proposed system performs classification of the vehicle at the sensor. It makes use of a simple sensor to avoid complications from sensitive and fragile components such as cameras and lasers. As a bonus, a simple system is also cheaper to produce and repair.

Current classification methods can be divided in two important groups, intrusive and non-intrusive. The first ones are embedded on the road surface, so it is necessary close the lane to install them, causing traffic disruptions, which increases installation and maintenance costs. In addition, they are usually adjusted to the pavement conditions, and calibrated so they can properly work. The second ones are mounted overhead on roadways or roadside. However, these are more sensitive to traffic and environments such as harsh weather and light conditions [5].

Neural Network and Deep Learning methodologies, have shown great performances in different areas. Such as, classification of image, video, text, signal or patterns. From the different models used, the most dominant ones are feed-forward, convolutional, and recurrent architectures [17].

Finally, was need to create a system that was able to collect the data and run it on the neural network. In order to obtain a real-time classification. In the end was decided to use an Field-Programmable Gate Array (FPGA), due to constrains in speed, cost, and energy consumption. This reconfigurable computing
Fig. 2. Vehicles categories according to the number of axles.

<table>
<thead>
<tr>
<th>Category</th>
<th># Axles</th>
<th>Silhouette</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIKE</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>PASSENGER</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2 AXLES</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3 AXLES</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4 AXLES</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5 AXLES</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6 AXLES</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7 AXLES</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8 AXLES</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

The rest of this paper is organized as follows. Background about vehicle classification in Section II. A description of proposed classification system in Section III. Neural network development is detailed in Section IV. Implementation, experimental results, and their discussing are in sections V, VI and VII, respectively. Conclusions and future work are in Section VIII.

II. BACKGROUND

A. Vehicle Classification

Over the years there have been several different approaches to the classification problem. Fig. 2 illustrates the silhouettes for the different classes of vehicles. Usually vehicles are characterized according to their height at the first axle, and number of axles. The classification mechanisms can be either intrusive or non-intrusive.

The most popular systems in intrusive methods are Inductive Loop Detectors. These sensors consist of many loops of wire, creating an inductor sensitive to ferromagnetic material. Because different vehicles are constructed with different materials and are disposed in different configurations, they create distinct magnetic profiles, which can be used to do the classification. However, the accuracy was not the best, something that was improved with artificial neural network assistance. They were able to achieve classification rates of about 92% with the implementation of a pattern recognition network [12].

Other alternatives, based in Wireless Sensor Networks (WSN) can be described in [9] and [4]. These are normally composed by sensor in the roads like magnetometer, accelerometer and a roadside access point. The magnetometers are used to estimate vehicle speed, while the accelerometers are used to detect the axles and calculate the axle spacing, the access point can be used to synchronize, send and receive data or made calculus. Typically, these approaches apply the information gathered from the different sensors to algorithms that allow it to compute the data needed for classification.

Fig. 3. Typical axle sensors on a road: pneumatic (left) and inductive (right).

From the group of non-intrusive the are some methods that are less common but are worth mention. A radio based system which leverages the attenuation of radio signals and uses machine learning algorithms, k-Nearest Neighbor (kNN) and Support Vector Machines (SVM), to classify the vehicles [6]. However it can only be used on one-way roads, unlike the proposed method in this work. An infrared and Ultrasonic based system which use a dynamic Bayesian network to classify the data features, that are harder to install. This sensors are placed over the road like a public street light.[11].

Other methods mention using acoustic sensor to extract features used by a kNN classifier [20] or a multi-modal sensing approach using magnetometers, electromagnetic RF field, and acoustics to classify [8]. Like the proposed method features extracting are an essential part, and the end results depend on it.

Fig. 3 illustrates the two most common methods to determine the number of axles, usually installed on roads: pressure pipes (left) and magnetic loops (right).

B. Neural Network Classification

Nowadays, the most common methods are based in video or image processing, from traffic cameras installed next to the roads. There are different studies, some use convolutional neural network [22], others use 3D point cloud and machine learning, kNN and SVM [1], or a more featured based approach [16]. With the current developments of neural networks and image processing, these methods are increasingly accurate, however nighttime vehicle classification is an harder task, due to the low luminosity.

Vehicle headlights present a possibility that is worth exploring, to be used as a classification method. The methods proposed in [21] consists of segmentation, detection, tracking, pairing, and classification of headlight on an image of two or four wheeled vehicles. However, this method, like all other similar ones found, are based on image processing, which requires a lot of computing power and energy something not ideal for real-time vehicle classification systems. The believe that measuring the light intensity emitted is a valid alternative will be explored in this work.

There are already studies showing that is possible to implement a Neural Network accelerator based on FPGA [7]. Other example is the implementation of a feed-forward neural
network that can realize real-time electro-acupuncture analysis [18].

III. PROPOSED CLASSIFICATION SYSTEM

The proposed classification system exploits the intrinsic features of the different vehicles, such as aerodynamics, shape, mass, type of suspension system, number of axles and their distance, engine vibration, and load, to distinguish them. Such features also impact how a vehicle responds to a certain type of pavement.

Inside a vehicle, the response can be assessed through sensors attached to the body, however, outside the vehicle, measuring such variation is not trivial. The proposed solution adopted using the measurements from the vehicle’s headlights. The headlights in a vehicle are tightly attached to its body, it means that the vibrations of the body are transmitted to its light beam. In other words, it modulates the perceived light. In layman terms, a motorbike has smaller mass than a truck, therefore it will bounce faster than the truck, meaning that they’ll produce different light variations. Fig. 4 illustrates the elements of a car’s suspension system. Different vehicles have different suspension mechanisms, with different responses. To increase the response of the physical system, a hump or a sleeping policeman can be installed on the floor to trigger a more pronounced response of the vehicle’s suspension system.

An acquisition system was created that uses a luminescence sensor capable of performing the measurements at a specific sample rate. The signal obtained, as well as its spectrum, was analyzed, looking for patterns capable of identifying different vehicles. However, this type of analysis is difficult to be done empirically due to very subtle differences between the signals. To assist in this process, a neural network was used as a non-supervised classification method for the type of vehicle.

A. Specifications

The process of data acquisition is a fundamental part of this project, as it needs to be able to observe vehicles circulating. The projections were made assuming that the vehicle does not exceed 200km/h, which is approximately 55.5(5) m/s or 18ms to travel 1m. Considering that 1k samples per meter, or 1 sample per millimeter, is sufficient to describe the signal, then the sample period should be 18 us.

Nyquist Theorem states that, to avoid aliasing, the sampling frequency (fs) should be at least twice the highest frequency (B) contained in the signal. With a sample rate of $2B$ samples/s the Nyquist frequency must be $fs > 2B$, which is equivalent to $B < \frac{fs}{2}$, so if $B = \frac{1}{T}$ we have a Nyquist frequency less than 111.1kHz. Therefore, the sensor used in this work must have a rise and fall time of less then 9us.

It is necessary to determine the quantization wordlength. One of its constraints is the resolution of the ADC and its sampling time. Usually higher resolutions take longer to do the conversion. However, a bigger resolution would allow to enhance the classification results.

B. System Architecture

Fig. 5 presents the high-level representation of the proposed system for real-time classification of vehicles. The light beam from a vehicle's headlights is measured using a phototransistor. The analog voltage is then converted to digital, with an Analog-to-Digital Converter (ADC), and then processed at the Digital Signal Processing (DSP) stage to extract its features. The features are then passed to the neural network block to perform classification of the signal.

IV. SIGNAL FEATURE EXTRACTION

Two similar signals may exhibit many different characteristics, or features, therefore, to facilitate the classification, the DSP block produces a set of features from an acquired signal. Some of the features passed to the Neural Network are described below. Some features are composed by several sub-features, for example for different frequency bands. The DSP block produces a total of 80 features.

A. Peak Detection

A signal peak indicates a maximum value before it starts decreasing. For an oscillatory signal, it is possible to observe several peaks. On a damped signal, such as a vehicle’s suspension, the peaks tend to become attenuated over time. The purpose of this feature is to detect peaks with higher prominence given a given threshold. The most important information, for each detected peak, is its prominence, width and position. After some experiments, it was chosen to save only 8 points to obtain the best possible result. In Fig. 6 it is possible to see an example of the detected peaks.

B. Autocorrelation

The autocorrelation provides observations of a signal over its delayed replicas. It helps to identify oscillatory patterns. Fig. 7 illustrates the output of the computation of the autocorrelation for an input signal.
### C. Power Spectral Density Estimate

The purpose of this feature is to detect the total power in some predefined frequency bands. Fig. 8 shows the power for different frequencies in the signal’s spectrum.

The outputs of the DSP block are summarized in Table I.

### D. Welch’s Power Spectral Density Estimate

The purpose of this feature is to detect Spectral peaks based on the Welch’s method that consists of dividing the time series data into (possibly overlapping) segments, computing a modified periodogram of each segment, and then averaging the Power Spectral Density (PSD) estimates, resulting in the Welch’s PSD estimate. Through experimentation, it was observed that 10 points provided the best results Fig. 9 it is possible to see an example of the Welch’s PSD estimate. In this figure it is also possible to identify the 10 selected points.

### V. NEURAL NETWORK-BASED CLASSIFIER

The most adequate, and usual, solution for a Pattern Recognition Network, a feed-forward network that can be trained to classify inputs according to target classes.

One problem of neural networks is the need to divide the data into three subsets. The first subset is the training set, which is used for computing the gradient and updating the network weights and biases. The second subset is the validation set, the error on this set is monitored during the training process. This error normally decreases at the initial phase of training, as does the training set error. However, when

---

**Table I**

<table>
<thead>
<tr>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean Value</td>
</tr>
<tr>
<td>Quadratic mean</td>
</tr>
<tr>
<td>Peak prominence</td>
</tr>
<tr>
<td>Peak width</td>
</tr>
<tr>
<td>Peak position</td>
</tr>
<tr>
<td>Spectral position</td>
</tr>
<tr>
<td>Spectral amplitude</td>
</tr>
<tr>
<td>Covariance amplitude</td>
</tr>
<tr>
<td>Covariance position</td>
</tr>
<tr>
<td>Power band</td>
</tr>
</tbody>
</table>

---

Fig. 6. Segmented input signal, including peak detection marks.

Fig. 7. Autocorrelation estimate for a segmented input signal.

Fig. 8. Power spectral density estimate.

Fig. 9. Welch power spectral density estimate.
the network begins to overfit the data, this error increases. The network weights and biases with the best results of the validation error will be saved. The third subset is the testing set, and is not used during training process, but it is used to compare different models and to plot the test set error during the training process. The best results for the Neural Network were obtained when using 70% of the samples for training, 15% for validation and 15% for testing.

A simple pattern recognition network from the Matlab neural network toolbox [10] was chosen to be used: a feed-forward neural network with back-propagation.

In the process of setting up the network, some parameters had to be defined. It was necessary to choose the function used to train and measure the network performance. In both cases the default Matlab function were used, trainscg and crossentropy respectively. A Tansing is used in the hidden layers and a Softmax is used in the output layer.

It was also necessary to find out which size and number of hidden layers had the best performance. The dataset into 5000 different sets, training, validation and test, randomly. These sets were used to test different network configurations and the means of performance were used to discover the best possible configuration. It was found that the optimal one contains two hidden layers, the first with 10 neurons and the second with 1, as seen in Fig. 10, obtaining a success rate of 79.70%.

VI. IMPLEMENTATION

A. Luminescence Sensor

The ability to measure the intensity of vehicles beam lights, is a key element of this work. As such, choosing the right sensor is a crucial task. Therefore, a thorough analysis of the sensors used in this kind of assignments was done, taking into account the required sampling time, and sensibility.

Of all the different types of sensors, the most commonly used ones are: photodiodes, phototransistors and photoresitors. Each one has its own strengths and weaknesses. Table II summarizes their characteristics in terms of light sensitivity, operating bandwidth and cost. The cost is relative to an average CMOS camera, frequent in image processing systems.

The phototransistor was selected because it offers the best tradeoff in terms of features and cost. At the end the SFH 309 [15] from OSRAM was chosen. Its ability to detect visible and part of infrared radiation, with a wavelength between 380 nm to 1180 nm, as well as the short and fall time, about 9us, were meaningful factors in this decision.

![fig10](image10.png)

**Fig. 10.** Topology of the Neural Network to implement the classifier.

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Sensitivity</th>
<th>BW</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Photo-resistor (LDR)</td>
<td>100uW/cm²</td>
<td>Hz</td>
<td>very low</td>
</tr>
<tr>
<td>Photo-diode</td>
<td>5000uW/cm²</td>
<td>MHz</td>
<td>low</td>
</tr>
<tr>
<td>Photo-transistor</td>
<td>10000uW/cm²</td>
<td>kHz</td>
<td>low</td>
</tr>
</tbody>
</table>

**TABLE II**

Characteristics of the most common different photo-sensitive sensors.

![fig11](image11.png)

**Fig. 11.** Calibration test results.

Its spectral sensitivity relative to the visible radiation ranges from 18% at 400nm to 76% at 700nm, so it will works better for lower temperature lights (hot colors) than higher ones (cold colors). Furthermore the directional characteristic, have a sensitivity that falls considerably according to the angle of incidence, being 0.7 and 0.2 for an angle of 10 and 20 respectively. A potentiometer of 100k ohm was used to adjust the sensor’s sensitivity.

To be sure that the sensor was work working properly, a calibration test was done. A RW-3703-2, radiometric detector, together with a P-9710-2, optometer, both from Gigahertz-Optik, were used to measure light irradiance emitted by a flashlight at different distances. Comparing this value with that measured by the Arduino it is possible to understand if the sensor is well calibrated.

To reinforce these observations from Fig. 11, some errors were calculated between the two. Obtaining an average error of 0.0018, an mean absolute deviation of 0.0056, a mean square root of $4.5 \times 10^{-05}$, a mean percentage error of 0.4668% and an mean absolute percentage error of 2.04%. From these results it was concluded that the sensor is well calibrated and will not have a negative influence on the measurements.

B. Platform to Collect Training Data

It was necessary to choose a development board to build the acquisition system. Arduino [3] was selected as it is an easily programmable board that also allows connection of the required components.

By default, it takes about 100us for the Arduino to perform an analog-to-digital conversion. Therefore, it would be unable to meet the time requirements, specified, since a sample time of 18us was required. To reduce the sampling time, with
no significant loss of resolution, it necessary to modify the prescaler, changing the input clock frequency from 125kHz to 1000kHz.

Nevertheless, it was necessary to quantify these losses. Experiments were performed testing all the prescaler setups, measuring waves with different frequencies. It was observed that the wave frequency directly influenced the measured error, increasing it. However, this variation with rising clock frequency was not significant, therefore, it will not condition the collected samples. Due to this hardware limitation the sampling period was readjusted, from 18us to 20us.

Afterward, due to Arduino memory limitations was necessary to find a solution to record all the collected information. As such, it was decided to use a SD Card, which allow storing large amounts of data. Besides that, a Real-Time Clock (RTC) module was used, to know the exact time when recording was started.

In addition, an interface with led and push-button was designed. Where the led was used to know when the system was ready to start or had finished recording and the push-button to control this action. In order for the led and the transistor to work properly, resistors were added to size the current that would pass through them. An Arduino shield was built to accommodate all the components and additional required electronics. The corresponding circuit schematic is in Fig. 12.

C. Logger Application

To implement the data acquisition system in the Arduino it is necessary to write the information in the SD Card very quickly and without any data loss due to buffer overruns. Since the default function to do this task presented in the Arduino was not fast enough the SdFat library was used which provides read/write access to FAT16/FAT32 file systems on SD/SDHC flash cards.

This implementation was based on the example program from the library called AnalogBinLogger. Some modifications had to be made in the interface to work as intended. To respect what was originally stipulated, 10 bits and 20us sampling period, some necessary configurations had to be made. After that was only necessary to do a code cleanup in order to remove what was not needed.

D. Classification System

To implement the Classification System, the data of the different characteristics of acquired signals were used in Matlab to train and generate the neural network.

Then a 32-bit floating point AXI4-Stream IP was implemented using Xilinx Vivado proprietary software with information acquired from the network generated by Matlab. Being the network composed of two hidden layers that use the tansig function and a final layer that use the function softmax. With an universal neuron capable of performing all the operations.

VII. EXPERIMENTAL RESULTS

The experiments were conducted on a street where there are frequent vehicles passing. To investigate the effect of the suspension system on the light beam, it was chosen a place where a crosswalk had a hump. A photo of part of the street where the experiments were conducted is in Fig. 13. In this figure includes the detail of the crosswalk with a hump.

A. Dataset

The next step was the collection of data and its respective labeling. A dataset was created, composed by two different types of vehicles cars and buses. It was acquired using a 50kHz sample rate, and each time series had about 9s.

In Fig. 14 and 15 is visible the average of different vehicles classes.

The differences are clear. On one hand, cars have a narrower wave when compared to buses, due to the different speeds with which the two types of vehicle pass the sensor. On the other hand, buses have fewer peaks and are further apart from each other when compared to cars.

In Fig. 16 and 17 are visible different vehicle waves to have a better perception of variations within the same class.

B. Hardware Resources

To finalize this study was necessary to implement the neural network in the FPGA and compare results with the Matlab.
implementation. The target platform was an ArduZynq TE0723 board with a Zynq XC7Z010-1CLG225C device from Xilinx.

The information about data offset, gain, ymin, weights and bias from Matlab code was used to do this implementation. In this design, it was built only a single neuron that performs the operations of all neurons and the forward propagation, plus a Finite State Machine (FSM) that controls the neuron switching among different neurons and logical layers of the neural network. Fig. 18 presents the block diagram with the architecture of the classifier based on a neural network.

It was chosen that this type of design be easier to scale the network with different topologies in a restricted resource environment. At the end, it was possible to achieved similar results, getting the same class classification output.

The resources used to implement the neural network are presented in the following tables.

In this implementation, to change the network size or increase the number of features, the only concern will be the memory available in the FPGA. Since, it was only used 5% the are a big margin for scalability.

### TABLE III
**SLICE LOGIC.**

<table>
<thead>
<tr>
<th>Type</th>
<th>Used</th>
<th>Util (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUTs</td>
<td>6223</td>
<td>35.36</td>
</tr>
<tr>
<td>Registers</td>
<td>1107</td>
<td>3.14</td>
</tr>
<tr>
<td>F7 Muxes</td>
<td>388</td>
<td>4.41</td>
</tr>
<tr>
<td>F8 Muxes</td>
<td>4</td>
<td>0.09</td>
</tr>
<tr>
<td>RAMB36E1</td>
<td>2</td>
<td>3.33</td>
</tr>
<tr>
<td>RAMB18E1</td>
<td>2</td>
<td>1.67</td>
</tr>
<tr>
<td>DSP48E1</td>
<td>42</td>
<td>52.50</td>
</tr>
</tbody>
</table>
C. Wordlength Optimization

The Matlab uses double precision in its code. However using smaller data types will allow the hardware to run the network faster and consume less resources, something that is really important when working with resource-constrained boards, as an FPGA.

Therefore, when using the Matlab C code generator, to obtain the C implementation, since it was possible to convert the data types, it was tried get three different network with single, double and fix-point precision.

Due to Matlab conversion errors, it was only possible to get the network working with single and double precision. Running both networks and comparing them, the error was in order of one millionth. It was concluded that the use of single precision will not affect network performance.

VIII. DISCUSSION

A. Neural Network Input Features

In the curse of the research of development of this method it was observed that some features have more impact than others. Whereas for this work such features were determined by experimentation, it was not obvious that to support other classes of vehicles such features wouldn’t matter. Moreover, depending on the variation required by the neural network to differentiate the different classes of vehicles, it may be possible to optimize the wordlength required in the computation of the DSP block. That is to say, if only few and extreme values of a feature contribute to make a decision on a class than there is no need to compute large wordlengths.

B. Comparison With a Software-Only Implementation

To show the advantages of running the neural network in a specific design hardware both implementations were compared. It was executed the matlab C code on the FPGA, Table IV, as well as, the hardware version using Vivado simulation environment, Table V.

Comparing both tables, the number of cycles from hardware implementation versus software implementation is a reduction of approximately 9000%.

IX. CONCLUSIONS AND FUTURE WORK

This work proposes a novel approach for classification of traffic in real-time, and presents its first implementation on a System-on-a-Chip (SoC) towards an IoT environment. So far, the proposed system allows for a reduced number of vehicle classes because of the location where the experiments were conducted.

Future work is planned to investigate the application to other classes of vehicles, optimize the implementation of the DSP block, and evaluate the implementation of other embedded architectures to minimize power consumption.

ACKNOWLEDGMENT

I would like to thanks my teachers Rui Antnio Policarpo Duarte and Horcio Cludio De Campos Neto for all the help throughout this work, and my family and friends for all the support.

REFERENCES


