







# Computational Intelligence for Analysis of Traffic Data

Hernán Winter<sup>1</sup> , Juan Serra<sup>1</sup> , Sergio Nesmachnow<sup>1</sup> ,  
Andrei Tchernykh<sup>2,3</sup> , and Vladimir Shepelev<sup>3</sup>

<sup>1</sup> Universidad de la República, Montevideo, Uruguay  
{hernan.winter, juan.serra, sergion}@fing.edu.uy

<sup>2</sup> CICESE, Ensenada, Mexico  
chernykh@cicese.mx

<sup>3</sup> South Ural State University, Chelyabinsk, Russia  
shepelevvd@susu.ru

**Abstract.** This article presents a system developed for the collection and analysis of traffic data obtained from traffic camera videos using computational intelligence. The proposed system is developed using the modern object detection library Detectron2. A pipeline-type architecture is used for frame processing, where each step is an independent, configurable functional module, loosely coupled to the others. The validation of the proposed system is performed on real scenarios in Montevideo, Uruguay, under different conditions (daylight, nightlight, and different video qualities). Results demonstrate the effectiveness of the system in the considered scenarios.

**Keywords:** Computational intelligence · Neural networks · Traffic data · Smart cities

## 1 Introduction

The growth of cities and traffic density have led to an increased demand for surveillance systems capable of automating traffic monitoring and analysis. The main goal of these automatic systems is to aid or even remove the human labor for vision based tasks that can be performed by a computer, providing regulators and authorities the ability to respond quickly to diverse traffic issues and situations.

Tasks such as vehicle counting and infraction detection are of great importance for Intelligent Transportation Systems [23]. Recently, computer vision based detection and counting algorithms [22] have shown to be more effective and outperform traditional traffic surveillance methods, such as methods using different kinds of sensors [12]. However, there are still many challenges and open issues in computer vision based vehicle detection and counting processes, caused by illumination variation, shadows, occlusion, and other phenomena.

In this line of work, this article presents a system applying computational intelligence (based on Artificial Neural Networks, ANN) to solve traffic analysis

problems using video recordings provided by surveillance cameras. The problems solved include vehicle detection, counting, classification, tracking, and detection of different types of traffic offenses, such as red light intersection crossing and parking vehicles in not allowed zones. The validation of the proposed system is developed using real traffic videos from the city of Montevideo, Uruguay.

The main contributions of the research reported in this article include: i) a methodology for the design of traffic analysis software systems using videos; ii) specific implementations of vehicle detection, counting, classification and tracking methods, and iii) the validation of the proposed methods on real scenarios.

The article is structured as follows. Section 2 presents a background on computational intelligence for image analysis. A review of the main related work is presented in Sect. 3. The technical aspects of the solution, including the proposed architecture, modules, and supporting libraries are described in Sect. 4. The experimental validation is reported and results are discussed in Sect. 5. Finally, Sect. 6 presents the conclusions and the main lines of future work.

## 2 Computational Intelligence for Image Analysis

This section describes the main concepts about the analysis of traffic data using computational intelligence.

### 2.1 Detection

One of the fundamental problems in computer vision is the task of assigning a label from a fixed set of categories to an input image. This task is known as image classification and is divided into three subtasks: segmentation, location, and detection.

The goal of semantic segmentation is to obtain a category for each pixel given an input image. It does not differentiate instances of the same object because each pixel in the image is classified independently. The classification and location task consists of classifying an image with a label that describes an object and drawing the box within the image around the object. The output in this task are a label that identifies an object and a box that indicates where that object is located. Object detection takes as input a set of categories of interest and an image. The goal of this task is to draw a box around each one of these categories, each time they appear in the image, and also predict the category. This problem is different from classification and localization since there can be a variable number of outputs for each input image.

Another task to consider is instance segmentation. Given an input image, this task seeks to predict the locations and identities of the objects in that image. Additionally, instead of simply predicting a region for each of those objects, this task seeks to predict a segmentation mask for each of those objects and to predict which pixels in the image correspond to each object instance.

In the last few years, computational intelligence and deep learning have led to successful results on a variety of problems, including image classification.

Among different types of deep ANNs, Convolutional neural networks (CNN) have been extensively studied [8]. CNNs assume that the input to be classified is an image. This assumption allows the network to be more efficient and to design architectures that greatly reduce the number of network parameters.

Solving the object detection problem involves determining all the regions where objects to be classified can be located. Given an input image, a Region Proposal Network (RPN) uses signal processing techniques to create a list of proposed regions in which an object can exist. This architecture class is named R-CNN. Given an input image, an RPN is executed to obtain the proposals, also called Regions of Interest (RoI). The main drawback of this approach is its very high computational demands. In practice, the network training is slow and needs significant memory. Fast R-CNN was proposed to mitigate these problems, working in a similar way to R-CNN. In terms of speed, Fast R-CNN has proven to be nine times faster than CNN in training time [7]. However, the computation time is dominated by the calculation of the RoI, which turns out to be a bottleneck. This last problem is solved in Faster R-CNN [17].

Finally, one of the most recent methods to solve the instance segmentation task is the Mask R-CNN architecture. Similarly to Faster R-CNN, this method follows a multi-stage processing approach. It receives the complete image, which is executed through a convolutional network and a learned RPN. Once the RoIs are learned, they are projected onto the convolutional vector. Then, instead of simply performing the classification and the regression of the regions of each RoI, the method additionally predicts a segmentation mask for each region, solving a semantic segmentation problem within each of the regions proposed by the RPN. The RoI is finally wrapped to the proper shape.

## 2.2 Tracking

Object tracking consists in the process of accurately estimating the state of an object -position,identity,configuration- over time from observations [14], thus generating a trajectory given by the position of the object in each frame. When several objects are located at the same time, the problem is called Multiple Object Tracking. In this scenario, the difficulty of the task increases considerably due to the occlusion generated by the interaction of the objects, which in turn may have similar appearances. On the other hand, conditions such as the speed at which the objects move, the lighting or that these change their appearance depending on the position, require that the tracking system must be robust, maintaining the object identifier in such situations.

In classical tracking methods, object features are extracted in each frame and used to search for the same object in subsequent frames [25]. This causes errors to accumulate in the process and if occlusion or frame skipping occurs, tracking fails because of the rapid change of appearance features in local windows. Thus, modern tracking methods apply two steps: object detection and data association. First, objects are detected in each frame of the sequence and then, detected objects are matched across frames. This paradigm is called tracking by detection [10] and it relies on the performance of the detection algorithm. Detected

objects are matched across frames using different approaches, including optical flow with mean shift of color signature, Earth mover's distance to compare color distributions, fragment-based features, and computational intelligence.

Another simple but effective method based on the tracking by detection approach is Intersection Over Union (IOU) [2]. This method requires a detection algorithm with a high rate of true positive results, as a detection is expected in each frame for each object to be tracked. It is also assumed that the detection of the same object in two consecutive frames present a great overlap of the intersection over the union (defined in Eq. 1), which is common for videos that present a high refresh rate.

$$IOU(a, b) = \frac{Area(a) \cap Area(b)}{Area(a) \cup Area(b)} \quad (1)$$

The advantage of the IoU method, in addition to its simplicity, is that it has lower computational cost than other methods. IoU can be integrated on other methods to achieve a more robust and accurate monitoring

### 3 Related Work

Several articles have proposed automated systems for the analysis of traffic data applying image processing and computational intelligence techniques. The most related to the research reported in this article are reviewed next.

Zhou et al. [28] studied the vehicle detection and classification problem applying deep neural networks. The You Only Look Once (YOLO) architecture was used for vehicles detection and post-processing was performed to eliminate invalid results. The Alexnet architecture was applied for classification, feature extraction, and fine-tuning. The YOLO network obtained similar precision than a Deformable Parts Model, while the Alexnet network using Support Vector Machines (SVM) outperformed other methods such as Principal Component Analysis and Absolute Difference in a public dataset.

Uy et al. [20] studied methods for identifying traffic offenses using genetic algorithms (GA) and the recognition of offenders through ANN. GA were applied to detect vehicles obstructing pedestrian crossings and to identify the location of license plates in images, while a ANN is used to recognize the license plate number. The license plates recognition accuracy was high (91.6% on 47 test images), but some license plates were not properly located due to the vehicle position respect to the camera. Zhang et al. [26] applied a Fully CNN with Long Short Term Memory for the vehicles counting problem. Compared to the state of the art, the proposed ANN architecture reduced the mean absolute error (MAE) from 2.74 to 1.53 on the WebCamT annotated dataset and from 5.31 to 4.21 on the TRANCOS dataset. In addition, the training time was accelerated by up to 5 times. However, the proposed ANN was not capable of handling long periods of information due to the large amount of memory required.

Dey et al. [5] applied CNN in a System-On-a-Programmable-Chip to analyze and categorize traffic, including the quality-of-experience variable to improve

predictions. A combination of transfer learning with re-training CNN models, allowed improving the prediction accuracy. Arinaldi et al. [1] applied computer vision techniques to automatically collect traffic statistics using Mixture of Gaussian (MoG) and Faster Recurrent CNN. Training and validation were developed on Indonesian road videos and a public dataset from MIT. Faster Recurrent CNN was best suited for detecting and classifying moving vehicles in a dynamic traffic scene, since MoG was weak for separating overlapping vehicles.

Chauhan et al. [3] studied CNN or real-time traffic analysis on Delhi, India. A YOLO network was used, pre-trained on the MS-COCO dataset and fitted with annotated datasets. The best trained model achieved a performance of 65–75% mean average precision, depending on the camera position and the vehicle class. This article provides the expected performance of YOLO models optimized using annotated data. The recent article by Zheng et al. [27] proposed TASP-CNN for predicting the severity of traffic accidents, considering relationships between accident features. The proposed method was successfully adapted to the representation of traffic accident severity features and deeper correlations of accident data. The performance of TASP-CNN was better than previous models when evaluated using data from an eight years period.

Our research group has developed research on detection on pedestrian movement patterns applying computational intelligence [4]. A flexible system was developed to process multiple image and video sources in real time applying a pipes and filters architecture to address different subproblems. The proposed system has two main stages: extracting relevant features of the input images, by applying image processing and object tracking, and patterns detection. The experimental analysis of the system was performed over more than 1450 problem instances, using PETS09-S2L1 videos and the results were compared with part of the MOTChallenge benchmark results. Results indicate that the proposed system is competitive, yet simpler, than other similar software methods.

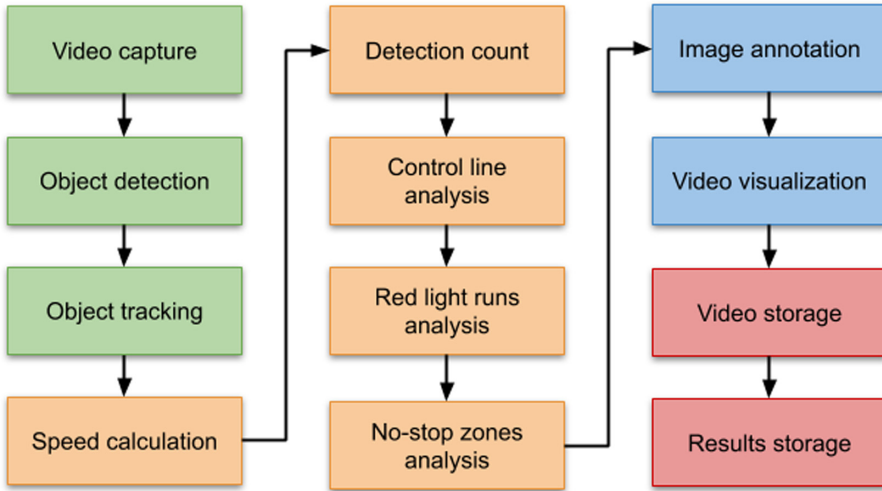
## 4 The Proposed System for Traffic Data Analysis

This section presents the implemented system for traffic data analysis, describing the function of each module and the input and output parameters.

### 4.1 Overall Description

The proposed approach is based on a modular architecture that implements an image processing pipeline [6]. The pipeline executes a set of tasks over input images (e.g., translation/rotation, resizing, etc.) to extract useful features. The modular architecture allowed for a progressive development process, starting from a few general modules and incorporating specific modules for relevant data.

Figure 1 shows the final architecture of the pipeline for traffic video analysis, consisting of twelve modules. The pipeline has 40 parameters that allow controlling different aspects of the processing in each module.



**Fig. 1.** The proposed pipeline for traffic video análisis (Color figure online)

Four main stages are identified: i) video capture, object detection and tracking (in green in Fig. 1), detection data analysis (in orange), results visualization (in blue), results storage (in red). They are described in the following subsections.

## 4.2 Video Capture and Object Detection

The goal of video capture is producing the frames used in the pipeline. A video stream (e.g., a local file or a webcam) is captured by a fast method using multi-threading parallel computing to read the video frames, using OpenCV utilities. Video capture initialize the cumulative data transfer object (DTO), used by all modules to read and write data, and stores several fields in the DTO, including *frame number*, *image* object, and *annotations*.

Then, the object detection process each frame to produce a bounding box, mask, score and class of the detected objects. This module is based on Detectron2 framework by Facebook [21], whose modular design allows using different state-of-the-art detection algorithms, such as Faster R-CNN, Mask R-CNN, or RetinaNet. The implemented module uses both synchronous and asynchronous detection, and adds different functionalities on top of the detection framework such as the possibility of defining regions of interest for the detection or filtering the classes of the detected objects. The output of the detection module is converted to the standard format used by the rest of the pipeline, so it might be replaced by other detection module without affecting the other modules in the pipeline. The output can contain bounding boxes or instance segmentation. The rest of the pipeline is compatible with both type of outputs and can take advantage of instance segmentation when available to compute more precise results when analyzing patterns.

### 4.3 Detection Data Analysis

Detection analysis includes five modules to extract information from data generated by previous modules in the pipeline.

The *speed calculation* module computes an estimation of the average speed of each detected object in recent frames, in pixels per frame (PPF) or pixels per second (PPS). The system stores the position of the center of each detected object in the last  $n$  frames ( $n$  is a parameter) and so the speed is given by the Euclidean norm of the first and last stored positions.

*Detection count* requires defining one or more counting lines on the video image. Based on a structure that keeps each detected vehicle as an object with several identifying properties, the counting module analyzes the vehicles that overlap with the counting lines. This analysis considers the intersection of the polygon of the mask or box with respect to the defined lines as an input parameter. If an overlapping is found, the vehicle information is updated with the lines it overlapped and the frame number in which it did so.

With *control lines analysis* it is possible to define a relationship between two lines, meaning that a vehicle should not cross both as doing so would be considered an infraction. This allows detecting different types of infractions that involves a vehicle circulating in a no-driving zone, e.g., a wrong turn or lane change near a corner. This module uses detected bounding boxes or masks to recognize if a vehicle overlaps with both lines in the relationship through the video.

The *red light runs analysis* module detects driving offenses of failing to comply with red light signal. The region where each semaphore is located is defined as an input parameter and each traffic light is associated with a line. The defined traffic lights are analyzed to determine their color frame by frame, applying an algorithm that transforms the cropped frame of the traffic light to Hue, Saturation, Value (HSV) color model.

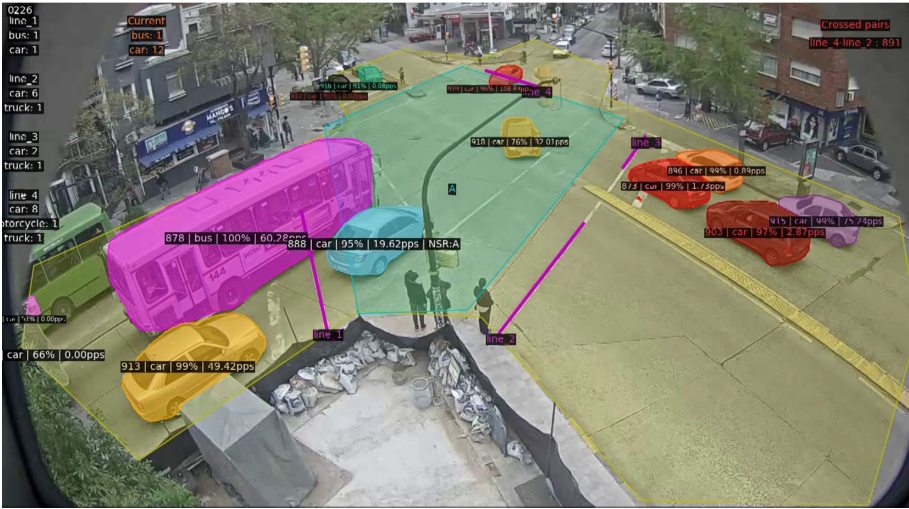
The *no-stop zones analysis* considers zones, represented by polygons, in which vehicles should not stop (or park). The module analyzes the bounding box or mask of those vehicles that intersects with the defined non-stop zone. If an intersection greater than a certain value is detected, then the average speed of the vehicle in the last  $n$  frames ( $n$  is a parameter) is considered. When the average speed reaches a value lower than one, the vehicle is considered in infraction and labeled as stopped.

### 4.4 Annotation and Results Visualization

The *annotation* module is responsible of modifying frames to show information generated by the previous modules. The modified frames will be part of the output video. Additionally, this module is in charge of drawing the detected objects, boxes, or masks as appropriate. Figure 2 presents an example of an annotated frame in one of the scenarios studied in this article.

The *video visualization* module is in charge of displaying the output frames as they are produced, using OpenCV to create a window and display the frames.





**Fig. 2.** Frame annotated with object masks, labels, text, lines, and polygons

## 4.5 Storage

The *video storage* module saves the frames in a file in a given path, considering the output format and FPS rate specified as parameters. Finally, the *results storage* module shows the information generated in each frame and the cumulative one, using a JSON-based logging system.

## 5 Validation Experiments

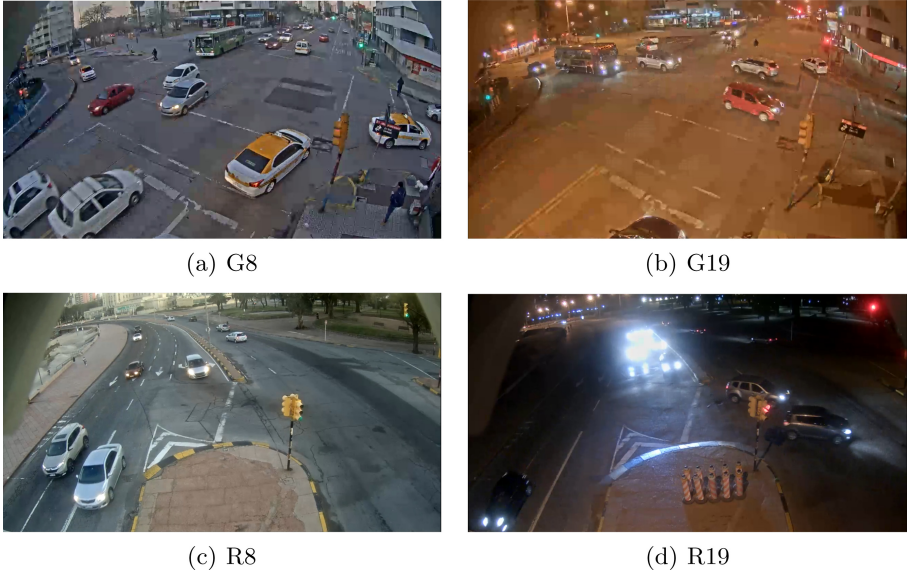
This section reports the validation experiments of the proposed system.

### 5.1 Case Studies in Montevideo, Uruguay

The experimental evaluation considered two case studies in Montevideo, Uruguay. The first case study corresponds to the intersection of 8 de Octubre and Garibaldi avenues, representing a classic intersection between two avenues in Montevideo. The second case study corresponds to the intersection between Rambla Wilson and Sarmiento Avenue. This case is relevant because it involves the avenue considered as the main traffic lane during rush hour.

The test dataset used consists of four videos taken by video surveillance cameras from the two studied locations. The camera model is AXIS P1365 Mk II and records up to 60 frames per second. Recordings were taken at two different times of the day, in the morning (8:00 AM) and in the evening (7:00 PM) to analyze the efficacy of the proposed system under different lighting conditions. Figure 3 presents sample images of the considered scenarios. In turn, Table 1 summarizes the main properties of the analyzed videos.



**Fig. 3.** Testing video scenarios**Table 1.** Properties of the test videos

<i>Reference</i>	<i>Format</i>	<i>Resolution</i>	<i># Frames</i>	<i>Rate</i>	<i>Duration</i>
G8	MP4	1280 × 720 pixels	2393	8 FPS	5 min
G19	MP4	1280 × 720 pixels	2398	8 FPS	5 min
R8	MP4	1280 × 720 pixels	5998	20 FPS	5 min
R19	MP4	1280 × 720 pixels	5997	20 FPS	5 min

## 5.2 Development and Execution Platform

The proposed system was developed using Python and Anaconda for project environment management allowing to install and maintain the required libraries easily. For the implementation, training, and execution of the presented ANN models, the Detectron2 framework [21], based on pytorch, was used. The tracking service was provided by a Node.js server [19] running an implementation of the Node Moving Things Tracker [15] library. OpenCV [13] was used for image and video manipulation and processing.

The experimental evaluation was performed on a virtual environment defined on a high-end server with Xeon Gold 6138 processors (40 cores and 80 threads per core), 8 GB RAM, a NVIDIA P100 GPU and a 300 GB SSD, from National Supercomputing Center (ClusterUY) [16]. Using this high performance computing platform, it was possible to dynamically reserve the resources needed for the batch jobs for the system execution and validation.

### 5.3 Metrics for Evaluation

Statistical measures were considered for the evaluation of the developed system. To account for the performance of stages that involve a binary classification, the standard metrics were applied: True Positive ( $TP$ ), which indicates the number of occurrences where the model correctly predicts the positive class; True Negative ( $TN$ ) is the number of occurrences where the model correctly predicts the negative class; False Positive ( $FP$ ) the number of occurrences where the model incorrectly predicts the positive class; and False Negative ( $FN$ ) indicates the number of occurrences where the model incorrectly predicts the negative class.

Metrics proposed by Sokolova and Lapalme [18] were applied to evaluate the performance of detection and classification algorithms, including:

- *Average Accuracy*: indicates the overall effectiveness of a classifier (Eq. 2).
- *Error Rate*: indicates the average per-class classification error (Eq. 3).
- *Precision*: reflects the percentage of the results which are relevant (Eq. 4).
- *Recall*: refers to the percentage of total relevant results correctly classified (Eq. 5).

$$\frac{TP + TN}{\frac{TP + TN + FP + FN}{n}} \quad (2)$$

$$\frac{FP + FN}{TP + TN + FP + FN} \quad (3)$$

$$\frac{TP}{TP + FP} \quad (4)$$

$$\frac{TP}{TP + FN} \quad (5)$$

Metrics proposed by MOTChallenge [11] were used to evaluate the tracking method. Special metrics are required for evaluating multiple object tracking:

- Identity Switches ( $IDSW$ ): indicates the number of occurrences where an already identified object is assigned a new identifier.
- Multiple Object Tracking Accuracy ( $MOTA$ ): is a global performance indicator of the tracker combining three sources of error. (Eq. 6, where  $t$  represents the frame and  $G_t$  the number of objects in frame  $t$ ).

$$MOTA = 1 - \frac{\sum_t (FN_t + FP_t + ID_{Sw})}{\sum_t G_t} \quad (6)$$

## 5.4 Results: Object Detection

For the evaluation of the object detection module, the configuration baseline of Detectron2 and the detection confidence threshold were taken into consideration.

The Detectron2 configuration baseline is a set of parameters which determines the type of ANN to be executed and the weight model. These baselines are part of the Model Zoo in Detectron2 [21]. The main properties of the selected baselines are presented in Table 2.

**Table 2.** Details of the configuration baselines of Detectron2 used in the experimental evaluation of object detection

	<i>R101-box</i>	<i>X101-box</i>	<i>R101-mask</i>	<i>X101-mask</i>
Backbone	R101-FPN	X101-FPN	R101-FPN	X101-FPN
Weights model	R101	X-101-32x8d	R101	X-101-32x8d
Using masks	No	No	Yes	Yes

The selected backbones used are ResNet-101+FPN (R101-FPN) which is a Faster R-CNN and ResNeXt-101+FPN (X101-FPN) which is a Mask R-CNN. The weight model R101 is an adaptation of the original ResNet-101 model [9] and X-101-32x8d is a ResNeXt-101-32x8d model trained with Caffe2 [24]. In turn, the detection confidence threshold allows discarding detected objects which classification score value is lower than a given value.

Tables 3 and 4 reports the results of the considered detection metrics for the G8 and G19 videos, respectively. These videos were selected as they account for a representative traffic flow of the city in rush hours in the morning (G8) and

**Table 3.** Classification metrics obtained with different settings in video G8

<i>Configuration</i>	<i>Average accuracy</i>	<i>Error rate</i>	<i>Precision</i>	<i>Recall</i>
R101-box-t03	0.93	0.07	0.91	0.73
R101-box-t05	0.87	0.13	1.00	0.48
R101-box-t07	0.78	0.22	0.93	0.23
X101-box-t03	0.93	0.07	0.89	0.75
X101-box-t05	0.88	0.12	1.00	0.49
X101-box-t07	0.79	0.21	1.00	0.24
R101-mask-t03	0.94	0.06	0.90	0.80
R101-mask-t05	0.93	0.07	0.97	0.72
R101-mask-t07	0.90	0.10	0.96	0.58
X101-mask-t03	0.93	0.07	0.89	0.79
X101-mask-t05	0.91	0.09	0.96	0.66
X101-mask-t07	0.91	0.09	1.00	0.61

in the night (G19). In these videos the camera angle is such that the North to South flow of vehicles occasionally occludes the vehicles in the West to East flow. Additionally, this scenario has a crossing with multiple traffic lights which makes the vehicles stop and accumulate producing interesting detection situations.

**Table 4.** Classification metrics obtained with the different settings in video G19

<i>Configuration</i>	<i>Average accuracy</i>	<i>Error rate</i>	<i>Precision</i>	<i>Recall</i>
R101-box-03	0.89	0.11	0.78	0.67
R101-box-05	0.82	0.18	0.89	0.40
R101-box-07	0.75	0.25	1.00	0.23
X101-box-03	0.84	0.16	0.70	0.53
X101-box-05	0.83	0.17	0.90	0.42
X101-box-07	0.73	0.27	1.00	0.21
R101-mask-03	0.87	0.13	0.67	0.67
R101-mask-05	0.85	0.15	0.67	0.60
R101-mask-07	0.86	0.14	1.00	0.49
X101-mask-03	0.89	0.11	0.77	0.70
X101-mask-05	0.87	0.13	0.92	0.56
X101-mask-07	0.87	0.13	0.96	0.53

Results in Tables 3 and 4 indicate that the proposed system had an overall average accuracy of 85% and indicate that the mask configurations computed better results than the box configurations in most cases (10 out of 12 instances). No significant performance differences between R101 and X101 models on box nor segmentation mask prediction were detected, but X101 had slightly better results in the night cases for segmentation.

## 5.5 Object Tracking

The main parameter to consider is the *tolerance*, i.e. the number of frames before the algorithm concludes that a tracked object is no longer in the sequence. The tolerance value depends on the frame rate of the recording. Values corresponding to half, one, and two seconds were considered in the study, as they represent time windows in which the probability of a identity switch among detections is low. That is, 4, 8, and 16 frames for G8 and G19 videos, and 10, 20, and 40 frames for R8 and R19. The performance of the object tracking module depends on the detection module. The X101-mask-05 detection setting was defined as basis for all tracking tests as it was the best performing detection configuration.

**Table 5.** Tracking tests results

<i>Configuration</i>	<i>MOTA</i>	<i>Configuration</i>	<i>MOTA</i>
G8-t04	0.81	R8-t10	0.86
G8-t08	0.83	R8-t20	0.87
G8-t16	0.82	R8-t40	0.89
G19-t04	0.47	R19-t10	0.10
G19-t08	0.49	R19-t20	0.14
G19-t16	0.46	R19-t40	0.13

Results in Table 5 show average MOTA scores of 85% for the daylight cases and significantly lower (30%) for the cases in the night. This indicates that the module performs significantly better in daytime scenarios. In 3 out of 4 cases, the configurations with one second of tolerance (8 frames in G8 and G19, and 20 frames in R8 and R19) had slightly better results than for nighttime scenarios. *MOTA* was mainly affected by *FN* values. In the tracking evaluation this occurs when an existing vehicle is not detected in a given number of frames, therefore it is not tracked. Based on this observation, improving the detection module in bad lighting conditions would also improve the tracker performance.

## 5.6 Pattern Analysis

For the evaluation of the counting module, the vehicle count resulting from the pipeline execution was compared to the number of vehicles obtained using manual counting. The performance of the method to count vehicles was measured using the detection and classification metrics presented above. To perform the evaluation, 30-second video segments were extracted from the four videos in the original test dataset. Four segments were considered for each video, thus having a total dataset of 16 segments from the two studied scenarios, eight taking place during the day and eight during the night.

Table 6 reports the results of the counting module evaluation, using the following configuration: the R101-mask-05 configuration was established for detection; the tracking module uses an IoU of 0.05, and a loss tolerance of eight frames for G8/G19 videos and 20 frames for R8/R19 videos.

**Table 6.** Counting test results

<i>Reference</i>	<i>Average accuracy</i>	<i>Error rate</i>	<i>Precision</i>	<i>Recall</i>
G8	0.92	0.08	0.92	1.00
G19	0.52	0.48	0.62	0.76
R8	0.87	0.13	0.91	0.95
R19	0.19	0.81	0.27	0.33

Results in Table 6 show an average accuracy of 89% for the daylight cases and 36% for the cases in the night. This indicates that the counting module performs better under good lighting conditions. In this case there is still room to improve the classification of the counted vehicles as the comparison of *precision* and *recall* shows that the main source of error are the *FP*. This means that the counting algorithm correctly counts a vehicle, but classifies it in the wrong class. Under bad lighting conditions the performance is poor, this is also caused by the large number of *FP*, which in this case happens because the counting algorithm counts not existing vehicles. The performance of this module can be mainly improved by using a more precise classification model in the detection module, while a better detection under bad lighting conditions would also improve the performance of the counting module. Improving the classification under poor lighting conditions is one of the main lines for ongoing and future work.

## 6 Conclusions and Future Work

This article presented the design and implementation of a system for the analysis of traffic data using computational intelligence techniques.

The proposed system was developed following a flexible pipeline-type architecture built over the modern object detection library Detectron2. The proposed design provides an efficient frame processing, by using independent, configurable functional modules, loosely coupled between them. This feature allows including new methods, modifying existing ones, and evaluate different alternatives and configurations.

The problems of object detection and tracking are solved using the Detectron2 framework and the Node Moving Things Tracker library, respectively. The information generated by these modules from the traffic videos allowed implementing a set of modules for the collection and analysis of traffic data.

The validation of the proposed system is carried out using real videos from two scenarios that include important streets of Montevideo, Uruguay, under different conditions (daylight, nightlight, and different video qualities). These recordings were taken in rush hours and show an interesting flow of vehicles.

Results demonstrate the effectiveness of the system in scenarios with proper lighting conditions. The detection results shown an overall average accuracy of 85%, and better performance using the mask models. In object tracking, the average MOTA scores were 85% in daylight. Results dropped to 30% in nighttime, indicating that improvements are required to deal with bad lighting conditions. Similarly, the counting module performed an average accuracy of 89% in daylight and 36% in nighttime.

The main lines for future work are related to improve the object detection module training the models with bad lighting conditions or bad weather annotated examples. In turn, the experimental evaluation of the proposed system can be extended to consider the analysis of red light runs and no-stop zones modules. Another interesting line of work is related to developing more sophisticated pattern detection methods to capture relevant events such as abrupt lane change or even traffic accidents. We are working on these topics right now.

## References

1. Arinaldi, A., Pradana, J., Gurusinga, A.: Detection and classification of vehicles for traffic video analytics. *Procedia Comput. Sci.* **144**, 259–268 (2018)
2. Bochinski, E., Eiselein, V., Sikora, T.: High-speed tracking-by-detection without using image information. In: 14<sup>th</sup> IEEE International Conference on Advanced Video and Signal Based Surveillance, pp. 1–6 (2017)
3. Chauhan, M., Singh, A., Khemka, M., Prateek, A., Sen, R.: Embedded CNN based vehicle classification and counting in non-laned road traffic. In: 10<sup>th</sup> International Conference on Information and Communication Technologies and Development (2019)
4. Chavat, J.P., Neschachnow, S.: Computational intelligence for detecting pedestrian movement patterns. In: Neschachnow, S., Hernández Callejo, L. (eds.) ICSC-CITIES 2018. CCIS, vol. 978, pp. 148–163. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-12804-3\\_12](https://doi.org/10.1007/978-3-030-12804-3_12)
5. Dey, S., Kalliatakis, G., Saha, S., Kumar Singh, A., Ehsan, S., McDonald, K.: MAT-CNN-SOPC: Motionless analysis of traffic using convolutional neural networks on system-on-a-programmable-chip. In: NASA/ESA Conference on Adaptive Hardware and Systems (2018)
6. Gilewski, J.: detectron2-pipeline: Modular image processing pipeline using OpenCV and Python generators powered by Detectron2. <https://github.com/jagin/detectron2-pipeline> (2019) 15 March 2020
7. Girshick, R.: Fast R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision (December 2015)
8. Gu, J., et al.: Recent advances in convolutional neural networks. *Pattern Recogn.* **77**, 354–377 (2018)
9. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. arXiv preprint [arXiv:1512.03385](https://arxiv.org/abs/1512.03385) (2015)
10. Leal-Taixé, L.: Multiple object tracking with context awareness. CoRR abs/1411.7935 (2014)
11. Leal-Taixé, L., Milan, A., Reid, I., Roth, S., Schindler, K.: MOTChallenge 2015: Towards a benchmark for multi-target tracking. [arXiv:1504.01942](https://arxiv.org/abs/1504.01942) [cs] (2015)
12. Lou, L., Zhang, J., Jin, Y., Xiong, Y.: A novel vehicle detection method based on the fusion of radio received signal strength and geomagnetism. *Sensors* **19**(1), 58 (2019)
13. Mahamkali, N., Vadivel, A.: OpenCV for computer vision applications (2015)
14. Moussy, E., Mekonnen, A.A., Marion, G., Lerasle, F.: A comparative view on exemplar ‘tracking-by-detection’ approaches. In: 2015 12th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), pp. 1–6 (2015)
15. Move-lab: Tracking things in object detection videos. <https://www.move-lab.com/blog/tracking-things-in-object-detection-videos> (2018) 15 Mar 2020
16. Neschachnow, S., Iturriaga, S.: Cluster-UY: collaborative scientific high performance computing in Uruguay. In: Torres, M., Klapp, J. (eds.) ISUM 2019. CCIS, vol. 1151, pp. 188–202. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-38043-4\\_16](https://doi.org/10.1007/978-3-030-38043-4_16)
17. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) *Advances in Neural Information Processing Systems* 28, pp. 91–99 (2015)



18. Sokolova, M., Lapalme, G.: A systematic analysis of performance measures for classification tasks. *Inf. Process. Manage.* **45**(4), 427–437 (2009)
19. Tilkov, S., Vinoski, S.: Node.js: Using javascript to build high-performance network programs. *IEEE Internet Comput.* **14**(6), 80–83 (2010)
20. Uy, A., et al.: Automated traffic violation apprehension system using genetic algorithm and artificial neural network. In: *IEEE Region 10 Technical Conference*, pp. 2094–2099 (2016)
21. Wu, Y., Kirillov, A., Massa, F., Lo, W., Girshick, R.: Detectron2. <https://github.com/facebookresearch/detectron2> (2019)
22. Yang, H., Qu, S.: Real-time vehicle detection and counting in complex traffic scenes using background subtraction model with low-rank decomposition. *IET Intel. Transport Syst.* **12**, 75–85 (2018)
23. Yang, Z., Pun-Cheng, L.: Vehicle detection in intelligent transportation systems and its applications under varying environments: A review. *Image Vis. Comput.* **69**, 143–154 (2018)
24. Yangqing, J., et al.: Caffe: Convolutional architecture for fast feature embedding (2014)
25. Zhang, K., Zhang, L., Yang, M.-H.: Real-time compressive tracking. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) *ECCV 2012*. LNCS, vol. 7574, pp. 864–877. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-33712-3\\_62](https://doi.org/10.1007/978-3-642-33712-3_62)
26. Zhang, S., Wu, G., Costeira, J., Moura, J.: Fcn-rlstm: Deep spatio-temporal neural networks for vehicle counting in city cameras. In: *International Conference on Computer Vision*, pp. 3687–3696 (10 2017)
27. Zheng, M., et al.: Traffic accident’s severity prediction: a deep-learning approach-based CNN network. *IEEE Access* **7**, 39897–39910 (2019)
28. Zhou, Y., Nejati, H., Do, T., Cheung, N., Cheah, L.: Image-based vehicle analysis using deep neural network: a systematic study. In: *IEEE International Conference on Digital Signal Processing*, pp. 276–280 (2016)