A NOVEL DEEP LEARNING BASED METHOD FOR DETECTION AND COUNTING OF VEHICLES IN URBAN TRAFFIC SURVEILLANCE SYSTEMS

J. J. Majin[1,], Y. M. Valencia [1], M.E. Stivanello[3], M. R. Stemmer[1,*], J. D. Salazar[2]

[1] Automation and Systems Department/UFSC - Florianopolis, SC, Brazil
[2] Mechanical Engineering Department, Labmetro/UFSC - Florianópolis, SC, Brazil
[3] Academic Department of Metal-Mechanics/IFSC, Florianópolis, SC, Brazil

KEY WORDS: Deep learning, DeepSORT, Object detection, Traffic monitoring, Vehicle counting, YOLOv4.

ABSTRACT:

In intelligent transportation systems (ITS), it is essential to obtain reliable statistics of the vehicular flow in order to create urban traffic management strategies. These systems have benefited from the increase in computational resources and the improvement of image processing methods, especially in object detection based on deep learning. This paper proposes a method for vehicle counting composed of three stages: object detection, tracking and trajectory processing. In order to select the detection model with the best trade-off between accuracy and speed, the following one-stage detection models were compared: SSD512, CenterNet, Efficiedet-D0 and YOLO family models (v2, v3 and v4). Experimental results conducted on the benchmark dataset show that the best rates among the detection models were obtained using YOLOv4 with mAP =87% and a processing speed of 18 FPS. On the other hand, the accuracy obtained in the proposed counting method was 94% with a real-time processing rate lower than 1.9.

1. INTRODUCTION

Through vehicle detection and counting, it is possible to establish traffic conditions, lane occupancy and congestion levels on highways. This information is a fundamental pillar in Intelligent Transportation Systems (ITS). ITS integrates several technologies into a single management system, such as automatic license plate recognition, traffic signal control systems, speed estimation and incident analysis (Zhang et al., 2011).

Most of the methods for vehicle detection and counting in ITS are based on hardware or software systems. Among hardware-based systems, inductive loops, piezoelectric sensors or ultrasonic detectors are commonly used. Although hardware solutions have a higher counting precision than software solutions, these sensors have limitations to obtain detailed information on the behavior of the vehicular flow, in addition to being intrusive and presenting high costs of installation and maintenance. On the other hand, software-based systems, especially video-based methods that perform image processing (computer vision) have started to stand out because it is an inexpensive, non-intrusive approach that have proven to be successful (Song et al., 2019).

The efficiency of computer vision systems for vehicle counting is largely related to good detection. Many researchers have used different methods for this task, among which are: (i) appearance-based detection methods, (ii) motion-based detection methods and (iii) detection models based on deep learning. Appearance-based methods perform detection using low-level features such as: symmetry, color, shape, texture, and edges. These features are generally extracted using Scale Invariant Feature Transform (SIFT) (Juan and Gwun, 2009), Histograms of Oriented Gradient (HOG) (Dalal and Triggs, 2005) and Haar-like (Mita et al., 2005), and then they are classified using machine learning methods such as Support Vector Machines (SVM) or Adaboost (Feature + Classifier). Motion-based methods perform detection when there are changes in the pixel values of the image, separating the foreground objects from the static background. These methods can be divided into three categories: frame difference methods or thresholding-based methods, optical flow method and background subtraction method. The models based on Deep Learning (Convolutional neural network-CNN[1]), perform detection in a more sophisticated way using deep architectures with the ability to learn complex features from images (Zhao et al., 2019).

Detection based on appearance and movement is most affected by problems associated with variations in scale and illumination. On the other hand, CNN-based methods present a certain degree of invariance to these problems and they are able to perform detection and classification of multiple object categories with high performance (Zhao et al., 2019). Object detection using CNNs can be categorized into two types: (i) regions proposal-based method (two stage) and (ii) proposal-free methods (one stage).

Proposal-based methods generate a set of proposal regions that may contain the objects of interest from an input image. Then the features of the regions are extended by a CNN. The classification of each of the regions in their respective classes is performed using a model based on machine learning. Among the most representative models of regions proposal-based are: R-CNN (Girshick et al., 2014), Fats R-CNN (Girshick, 2015), Faster R-CNN (Ren et al., 2015) and Mask R-CNN (He et al., 2017). Although these models present high precision in detection and location, their great disadvantage is that they require high computational costs that do not allow them to be implemented in practical real-time processing systems (Zhao et al., 2019).

On the other hand, proposal-free or one-shot (one-stage) methods are considered real-time detectors. These methods perform object detection as a global classification/regression problem,

* Corresponding author- marcelo.stemmer@ufsc.br

[1] CNN is one the most important and most successful methods of deep learning in the field of image analysis

directly mapping the bounding box and class probabilities of the feature maps generated by a single network. This approach entirely removes the region proposal step and subsequent pixel resampling steps, greatly reducing the processing time. Among the main one-stage methods are: SSD (Single Shot Multibox Detector) (Liu et al., 2016a), RetinaNet (Lin et al., 2017) and YOLO family detectors (You Only Look Once) (Redmon et al., 2016).

Due to the characteristics of the structure of one-stage methods, they present a low computational cost. However, the level of precision decreases depending on each model. Finding a balance between these two factors (precision and computational cost) is not a simple task, it strongly depends on the purpose of each implementation. Additionally, detection methods, especially in vehicle counting tasks, face many challenges, such as differences in perspectives, occlusions, illumination effects, and many more (Ciampi et al., 2018). Considering these factors, the main objective of this work is to propose a vehicle counting method able to minimize these problems, performing tracking and re-identification tasks on roads with high traffic flow. To do this, a detection model is first identified that presents a trade-off in terms of precision (mAP - mean Average Precision) and speed (FPS - Frame Per Second). Different one-stage detectors were compared: SSD512, YOLOV2 (Redmon and Farhadi, 2017), YOLOV3 (Redmon and Farhadi, 2018), YOLOv3-SPP (with spatial pyramid pooling operator (He et al., 2015)), and three recently proposed state-of-the-art detectors, CenterNet (Zhou et al., 2019), EfficienDet(D0) (Tan et al., 2020) and YOLOv4 (Bochkovskiy et al., 2020). This model is then used in the proposed counting method.

2. RELATED WORK

Vehicle counting using computer vision has been approached with different techniques. As mentioned above, the methods generally used for counting can be classified into three groups, feature-based methods, motion-based methods, and Deep Learning-based methods. Concerning the related works, the methods based on CNN have presented high performance in techniques for counting vehicles. In this context some works that perform the detection and counting of vehicles based on CNNs are presented.

In (Zhang et al., 2017) a technique is proposed that integrates a novel FCN-rLSTM network architecture to jointly estimate vehicle density and vehicle count by connecting Fully Convolutional Neural (FCN) networks with Long Short Term Memory (LSTM) networks in a residual learning fashion. The FCN makes the prediction of the pixels that the vehicles represent and LSTM learns the dynamic space-time of the scenario. Works such as (Lin and Sun, 2018) and (Asha and Narasimhadhan, 2018) perform vehicle counting using the YOLO detection model in a simple vehicular traffic scene. In these works, rules based on the coordinates of the bounding boxes in each video frame are proposed for counting using a virtual line. In (Hardjono et al., 2019) a comparison between classical image processing techniques and the YOLOv2 model for vehicle counting is presented. Results of this study conclude that the lowest counting error (-0.8) was obtained with YOLOv2.

In (Chen et al., 2018), a method for vehicles counting and tracking using the SSD detection model is presented. The center position of the vehicles was used to count each time they crossed a virtual line. The tracking was performed using the minimal Euclidean distance between centers for consecutive frames. A comparison between the SSD and GoogleNet models is presented in (Szegedy et al., 2015). These models are evaluated in terms of detection and counting accuracy for two classes: vehicles and people. The proposed algorithm uses the centers of the objects for their respective counting and tracking. The algorithm was evaluated on 5 videos of a one-way highway with low traffic flow, obtaining a counting accuracy of 95% and 83% for the two models, respectively.

(Dai et al., 2019) proposes a vehicle counting and tracking technique based on YOLOv3 and a Kernelized Correlation Filter (KCF). The KCF performs tracking with a matching method between bounding boxes provided by YOLOv3. This matching of the boxes is done for consecutive frames. The authors report an average counting accuracy of 90% in three different test scenarios; however with the increase of vehicles, the tracking accuracy begins to decrease. In (Song et al., 2019) a method is proposed for the problem in detecting and counting small vehicles, this method foregrounds the surface of the highway dividing it into a remote area and a proximal area. These areas are the inputs of the YOLOv3 model for the detection of the car, bus and truck categories. Finally, the ORB algorithm is used to obtain the number of vehicles with their respective trajectories. The average count accuracy was 93.2% in four low traffic flow test scenarios.

(Santos et al., 2020) proposes a system that uses YOLOv3 for object detection and DeepSORT for multiple object tracking. The authors use the identifier assigned by DeepSORT to count in a single category, reaching a 90% count accuracy in two low traffic flow test videos. The system was affected by identity changes of the same vehicle due to detection failures or occlusions. In (Mandal and Adu-Gyamfi, 2020) they proposed a Detection-Tracking based Vehicle Counting Framework. The authors combined different one-stage detection models (YOLOv4, EfficientDet, Detectron2, CenterNet) with object tracking algorithm models (SORT, KIoU, IoU, DeepSORT), with the aim of identifying the best combination that allows obtaining the better counting results. The proposed framework assigns an identifier to each vehicle based on the information from the tracking algorithms and the coordinates of the bounding boxes provided by the detection models. The experimental results of this study demonstrate that YOLOv4 and DeepSORT, Detectron2 and DeepSORT, and CenterNet and DeepSORT were the most ideal combinations for counting tasks. The performance of this framework was affected by occlusions and lower visibility that created identity changes and counted vehicles more than once.

3. METHODOLOGY

This section presents a description of the dataset created for the training and evaluation of the detection models, together with its evaluation metrics. Additionally, a description of the framework proposed for vehicle counting is made.

3.1 Dataset

The dataset is composed of 4300 RGB images captured on the highways of the city of Florianopolis-Brazil, at different times of the day. A Fujifilm FeniPix SL 300 digital camera was used to capture the images, with a resolution of 1280×720 pixels. To obtain the images, 9 different places of the city with dynamic vehicular flow of the classes of interest were considered. In detail, the images are captured at an average height of 4 meters with a general perspective of the highway, thus guaranteeing images with great variability, such as multiple orientations, different aspects, shadows, changes in lighting and different levels of vehicular congestion (Figure 1).

Figure 1. Some images from the dataset under different scenarios.

The dataset was manually labeled in five categories: car, bus, truck, van and motorbike with a total of 32672, 1574, 2818, 1605 and 4194 labels, respectively. One of the main dataset features is the vehicle size. This size varies from 40 pixels for small vehicles to 6992 pixels for large vehicles. This feature allows to establish the models that present limitations in the object detection with different sizes. Figure 2a shows some examples of the vehicle categories on this dataset and Figure 2b shows the number of labels per category used for training and test.

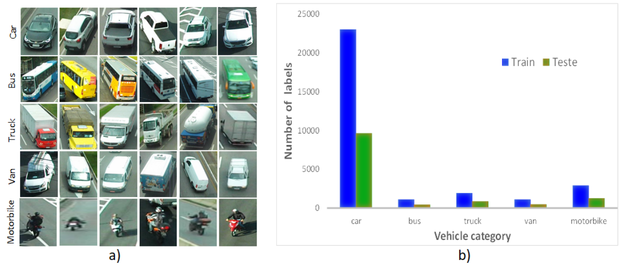


Figure 2. Dataset categories and number of labels.

3.2 Evaluation metrics

The precision, recall, F1 score, and mAP metrics were used to evaluate and compare the performance of the detection models. Precision refers to the percentage of predictions that are relevant instances. The recall measures the percentage of the model's ability to classify all samples that are positive within the dataset's ground truth. F1 score is the weighted average of precision and recall and allows to identify the model with the best balance between these two metrics.

The mAP is one of the main standard metrics used for the evaluation of different object detection models (Liu et al., 2020). The mAP is calculated based on the average of the AP (Average Precision) for all object categories. The AP is the area under the precision-recall curve on the IoU (Intersection over Union) that measures the overlap between ground truth and model prediction. Mathematically the definition of precision, recall, F1 score and mAP are shown in Equations 1 to 4, respectively.

$$precision(P) = \frac{TP}{TP + FP} \tag{1}$$

$$recall(R) = \frac{TP}{TP + FN} \tag{2}$$

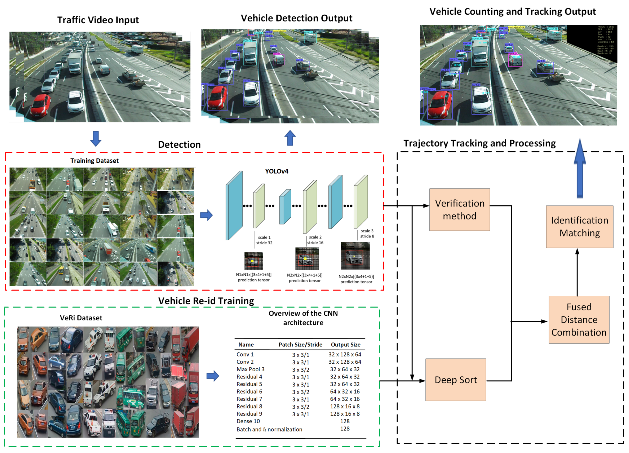
$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{3}$$

$$mAP = \frac{1}{N} \sum AP_i \tag{4}$$

Here, the TP (True Positive), FP (False Positive), and FN (False Negative) represent right detections, wrong detections and missed detections, respectively. N is the number of categories.

3.3 Proposed framework

Figure 3 shows the proposed framework for vehicle detection and counting. There are three main stages in this framework: object detection, tracking, and trajectory processing. Object detection is performed to obtain the bounding box and the classification of each object in the frame. Tracking is used to determine the trajectory of each object in the video sequence. The trajectory processing allows to obtain a result of the trajectories made by the objects from an analysis by regions.


Traffic Video Input
Vehicle Detection Output
Vehicle Counting and Tracking Output
Detection
Training Dataset
YOLOv4
scale 1 stride 32
scale 2 stride 16
scale 3 stride 8
N1xN1x[(3x4+1+5)] prediction tensor
N2xN2x[(3x4+1+5)] prediction tensor
N2xN2x[(3x4+1+5)] prediction tensor
Vehicle Re-id Training
VeRi Dataset
Overview of the CNN architecture
Trajectory Tracking and Processing
Verification method
Identification Matching
Fused Distance Combination
Deep Sort

Name	Patch Size/Stride	Output Size
Conv 1	3 x 3/1	32 x 128 x 64
Conv 2	3 x 3/1	32 x 128 x 64
Max Pool 3	3 x 3/2	32 x 64 x 32
Residual 4	3 x 3/1	32 x 64 x 32
Residual 5	3 x 3/1	32 x 64 x 32
Residual 6	3 x 3/2	64 x 32 x 16
Residual 7	3 x 3/1	64 x 32 x 16
Residual 8	3 x 3/2	128 x 16 x 8
Residual 9	3 x 3/1	128 x 16 x 8
Dense 10		128
Batch and ℓ_2 normalization		128

Figure 3. The proposed framework for detection and counting vehicles.

3.3.1 Object detection: This study used YOLOv4 to perform the detection. This model obtained the best results in terms of precision and speed based on the results of this research (section 4.1).

YOLOv4 represents a continuous improvement of the YOLO family. The detection principle in the YOLO models is the same, they divide the input image into an S×S grid and each cell in the grid is responsible for predicting the objects (Figure 4). If the center of the object is in a cell of the grid that cell is responsible for the detection of that object. Each cell of the grid predicts B bounding boxes with their respective confidence scores and simultaneously the probabilities of the categories of the objects (for each cell there is one probability by category). As the input image is divided into a grid, an object can occupy more than one cell, generating several duplicate bounding boxes for the same object. These redundant bounding boxes are filtered using the Non-Maximum Suppression (NMS) method (Rothe et al., 2014) which generates a single bounding box for each object. In the YOLOv3 and YOLOv4 versions, three grids of different sizes (13×13, 26×26 and 52×52) are used for detection at multiple scales.

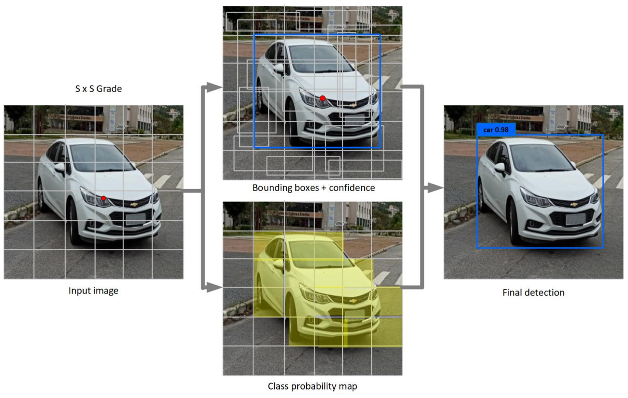


Figure 4. The detection principle of the YOLO model.

The YOLOv4 CSPDarknet53 network is mainly based on the Darknet-53 network of YOLOv3 with the integration of new convolution blocks, Spatial Pyramid Pooling (SPP), Path Aggregation Network (PANet) and Cross Stage Partial Connections (CSP). The SPP block is used to increase the receptive field and improve the extraction of the most significant features at various levels through the combination of space pyramids. The PANet block allows the bottom-up and top-down feature maps to be concatenated before entering the convolution blocks. CSP is integrated to enhance the variability of the learned features within different layers, by separating the feature maps by means of a partial dense block and the partial transition layer. Additionally, YOLOv4 presents two packages that integrate a set of techniques that facilitate detector training, Bag of Freebies (BoF) and Bag of Specials (BoS). These techniques include: Complete Interaction over Union (CIoU) and Distance Intersection over Union (DIoU). CIoU is used to improve the precision in the bounding box regression and DIoU is used as an NMS method to eliminate redundant bounding boxes based on the minimum distance between boxes.

YOLOv4 provides an output vector with six values <*x, y, w, h, class, confidence*> where (*x, y*) denotes the location of the center of the bounding box in the image, (*w,h*) represents the width and height of the predicted box, class and confidence are the type and score of the detected object, respectively.

3.3.2 Tracking: To achieve successful vehicle tracking, data association methods are required to relate the vehicles detected in the current frame to those in the previous frames. In this work, two methods were implemented to perform the tracking, the DeepSORT method and a verification method.

The DeepSORT method integrates a motion estimation based on Kalman filtering with deep appearance features to track multiple vehicles across video frames. This method maintains the vehicle's identities regardless of scale and orientation changes. Because DeepSORT was originally designed for human identification (Wojke and Bewley, 2018), it was necessary to train the CNN architecture (wide residual network) on a new Vehicle Re-id (VeRi) dataset available in (Liu et al., 2016b). This dataset was created with vehicular traffic videos captured by 20 surveillance cameras.

The DeepSORT method receives the object bounding box information provided by YOLOv4 and obtains the following parameters as output: $< x, y, w, h, \dot{x}, \dot{y}, \dot{w}, \dot{h} >$. Where $\dot{x}, \dot{y}, \dot{w}, \dot{h}$ are the first-order derivatives of the x, y, w, h parameters in consecutive frames provided by the Kalman filter. These coordinates, together with the appearance features from the pixels inside the bounding boxes, form the observations required to update the vehicle's state, where each one is associated with an Id.

The verification method is designed to confirm the identification of the vehicles at the scene. From the bounding boxes and Ids provided by DeepSORT, the center (cx, cy) of each vehicle is calculated. This information is stored in a vector $<(cx_0, cy_0, Id_0), ...(cx_n, cy_n, Id_n)>$, where n is the number of vehicles detected in the frame. Then, the Euclidean Distance between each center detected in the current frame with the center associated with the same Id in the previous frame is calculated as shown in the Equation 5.

$$Distance = \sqrt{(x^c_{id} - x^p_{id})^2 + (y^c_{id} - y^p_{id})^2} \tag{5}$$

where x^c_{id}, y^c_{id} refer to the coordinates of the center point of the current vehicle and x^p_{id}, y^p_{id} the coordinates of the center point of the vehicle in the previous frame with the same Id. It is verified if the distance is less than the threshold[2] ($h = 90$). If this condition is satisfied it is possible to confirm that the Id in question belongs to the same vehicle. On the contrary, the Euclidean distance of the evaluated center with all the centers detected in at least five past frames is calculated. Here, two situations can arise: situation 1, that there is no relation $Distance < h$ with any past center, in this case the detection of a new vehicle in the scene is confirmed; situation 2, that there is a distance less than the threshold h between two centers, in this case the same Id of the corresponding center is assigned. This last situation minimizes problems associated with detection failures by the YOLOV4 model or occlusions between objects.

3.3.3 Trajectory processing: The purpose of trajectory processing is to monitor and quantify vehicle categories in relation to their trajectories at the scene. For this, a set of input regions (Ir) and output regions (Or) were defined. These regions are created based on the geographical characteristics of the scene and are constant throughout the processing. From these regions it is possible to determine the displacement (dis) made by each vehicle in the scene, as shown in Figure 5.

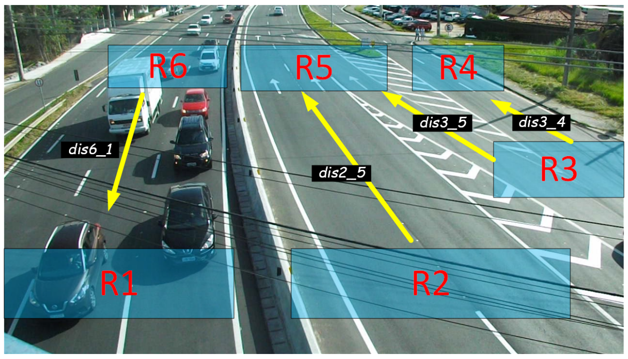


Figure 5. An example of the definition of regions for monitoring the movement of vehicles.

[2] This threshold was defined based on experiments performed to determine the minimum distance associated with the same vehicle in consecutive frames.

The first step in the trajectory processing consists of identifying the output region of each vehicle, for this, the following condition is evaluated:

$$Ir = \{Id \notin D \mid \forall_{Id=n} \in 1...n : Ptl \leq Pc \leq Pbr\} \quad (6)$$

where Pc represents the center of the vehicle, Ptl and Pbr represent the top left and bottom right points of the region R, respectively and D is a dynamic list that stores the tuple (Id, Ir) each time a vehicle is identified in an input region. The condition $Id \notin D$ ensures that the Id vehicle is stored only once in the input region where it was initially identified, preventing the same vehicle from being counted more than once.

The next step in trajectory processing is to identify the output region of the vehicles based on the evaluation of the following condition:

$$Or = \{Id \in D \mid \forall_{Id=n} \in 1...n : Ptl \leq Pc \leq Pbr\} \quad (7)$$

In this case, only the vehicles that were identified in an input region are evaluated, that is, that meet the condition $Id \in D$. If a vehicle is detected in an output region, the displacement (dis_Ir_Or) made by the vehicle is determined with the Ir and Or information. Once a displacement has been defined, the vehicle counter is increased by one in their respective class and type of displacement.

4. EXPERIMENT AND ANALYSIS OF RESULTS

This section consists of three parts. The first part evaluates the CNN-based one-stage detection models on the created dataset. The second part presents the analysis of the algorithm results for vehicle counting in four test videos. Finally, a runtime analysis of the framework is performed. All models were run on a computer with Intel Core i5-9300H 2.4 GHz quad-core CPU with 8 threads, an NVIDIA GeForce GTX 1060 Intel UHD Graphics 630 GPU and 4 GB of RAM. Among the software packages installed on this computer include Windows 10, Python 3.7, TensorFlow 2.0, PyTorch 1.5, OpenCV 4.4, and CUDA 10.2.

4.1 Object detection experimental

One of the main aims of this paper is the selection of a detection model with a trade-off in terms of precision and speed that can be used in the proposed method. For this, the following detection models were compared: SSD512, CenterNet, Efficiedet-D0, the YOLO family models, including YOLOv2, YOLOv3, YOLOv4 and YOLOv3-SPP.

The CNNs require a large amount of data for training before reaching a degree of generalization. Commonly, CNNs are trained on generic dataset such as COCO (Lin et al., 2014) or PASCAL VOC (Everingham et al., 2010) and are then adjusted for another domain of interest. This process is known as transfer learning, where the features learned by the CNN of a source domain (generic dataset) are transmitted to a different destination domain (custom dataset) (Yosinski et al., 2014). In this context, transfer learning was used to train the detection models from the weights obtained on the COCO dataset.

The training of the models was performed for 250 epochs using a learning rate of 0.001 with a batch size of 16. The input size of the models was set to 512 × 512 pixels and the sizes of the anchor boxes were adjusted based on the dataset. To avoid overfitting problems, data augmentation techniques were used during the training, including vertical flip, horizontal flip, changes in brightness and variations in scale (zoom).

The results obtained by each detection model are shown in Table 1. The most recent state-of-the-art detectors CenterNet and YOLOv4 obtained the best mAP, with 85.2% and 87%, respectively. YOLOv4 was the best detector among the models with 2% and 3% gain relative to CenterNet and YOLOv3. YOLOv3-SPP obtained a mAP=83%, 2% lower than YOLOv3. On the other hand, YOLOv2 provided the poorest results (mAP=68%) among the YOLO family models. SSD512 reached a mAP=70.56%, beating EfficienDet-D0 by 36%, which was the model that obtained the lowest percentage in all metrics, with a mAP=45%. In relation to the processing speed, all the models on average reached from 14 FPS to 18 FPS, except for YOLOv2 with 40 FPS.

Models	P	R	F1	mAP	FPS
SSD512	85.1	78.0	81.3	70.56	14.3
EfficientDet (D0)	89.4	40.6	55.9	45.0	15.0
CenterNet	80.5	**95.2**	**87.2**	**85.2**	14.0
YOLOv2	82.9	72.7	77.5	68.6	**40.0**
YOLOV3	84.2	93.9	88.8	84.6	14.0
YOLOv3-SPP	**87.7**	92.3	**89.9**	83.0	13.5
YOLOv4	**85.8**	**95.1**	**90.2**	**87.0**	**18.0**

Table 1. Result of detection in custom dataset.

From Table 1, it is also possible to confirm that YOLOv4 produced a high number of true positives and few false positives/negatives, achieving a good balance between the precision (85.8%) and recall (95.1%) metrics, as well as the highest F1 score (90.2%). These results are confirmed in Figure 6, where the precision/recall curves obtained by all the detectors are shown. As can be seen, the YOLOv4 curve (red) remains higher than the other curves. With the above, the YOLOv4 model is selected to be used for object detection in the vehicle counting method.

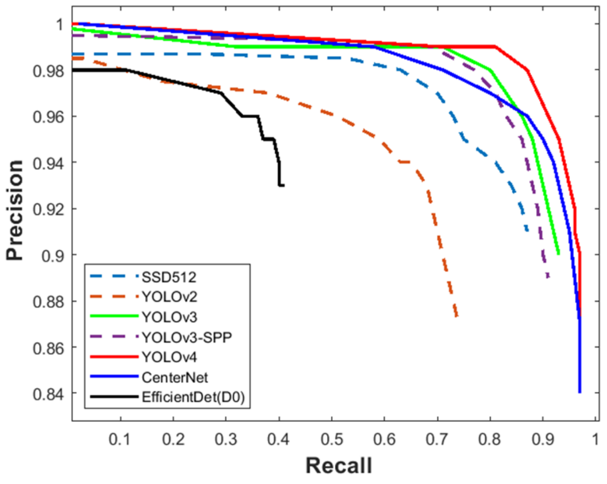


Figure 6. Comparison of precision/recall curves in custom dataset.

4.2 Counting Vehicle results

To validate the proposed framework, four video clips with a duration between 50 to 80 minutes were captured in four different places in the city of Florianópolis (Brazil). The resolution of each video was 1280×720 pixels at 30 FPS. Figure 7 shows the scenarios with their respective input and output regions. These scenarios feature straight-line and T-interception movements.

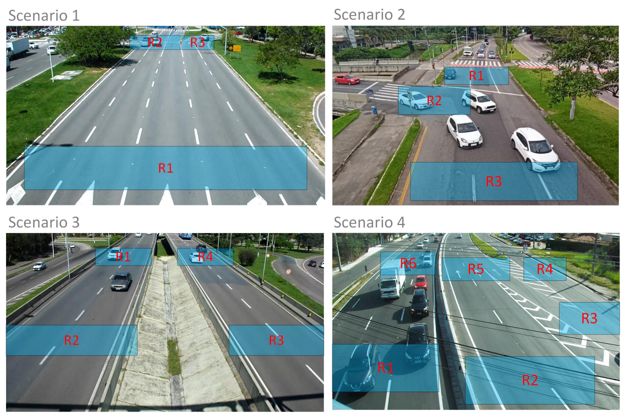


Figure 7. Test scenarios to validate the framework.

Based on the experiments performed, some parameters were defined for YOLOv4, the size of the input image was set to 416×416 pixels, and the score and IoU detection thresholds were set to 0.6 and 0.5, respectively. The metric used to measure the accuracy of the framework is shown in Equation 8.

$$accuracy = 1 - \frac{|GT - CA|}{GT} \tag{8}$$

where GT refers to the actual count using human vision and CA refers to the count performed by the framework. The vehicle count results are shown in Tables 2 to 5, corresponding to scenarios 1 to 4 of Figure 7, respectively.

Category	R1 - R2 (CA / GT)	R1 - R3 (CA / GT)	Accuracy (%)
Car	545 / 549	1027 / 1029	99.50
Bus	18 / 18	2 / 1	50.00
Truck	8 / 8	23 / 22	97.70
Van	11 / 10	44 / 45	93.85
Motorbike	50 / 51	218 / 223	97.85

Table 2. The framework results of scenario 1.

Category	R1 - R3 (CA / GT)	R2 - R3 (CA / GT)	Accuracy (%)
Car	1024 / 1077	962 / 953	96.75
Bus	1 / 2	12 / 12	75.0
Truck	39 / 38	24 / 22	93.95
Van	53 / 55	23 / 23	100
Motorbike	128 / 162	192 / 168	82.35

Table 3. The framework results of scenario 2.

As shown in Tables 2 and 3, the average accuracy obtained was 97.5% and 89.61% for scenarios 1 and 2, respectively. Bus accuracy was the lowest among the five vehicle categories. It was probably caused by the lower number of buses detected during this time period (19 in scenario 1 and 14 in scenario 2) compared to other vehicles, such as, for example, the car category with 1578 and 2030 samples for each scenario. Another factor that affected the results was the vehicle detection failures due to occlusions or when the vehicles present characteristics of similar appearance. This may result in YOLOV4 performing incorrect detection, especially when the view of vehicles changes rapidly.

In scenario 2, a greater number of occlusions were presented due to the structure of the highway and the camera angle. In this scenario some vehicles were not detected in their original region, this caused an error in the count, especially for the car and motorbike categories. An example of this case is presented in R1, in which vehicles that were not detected in this region were mostly detected in R2 as shown in Table 3.

Category	R1 - R2 (CA / GT)	R3 - R4 (CA / GT)	Accuracy (%)
Car	2415 / 2429	1016 / 1028	99.10
Bus	25 / 28	28 / 30	91.25
Truck	148 / 163	48 / 50	93.35
Van	139 / 142	51 / 49	96.85
Motorbike	305 / 315	142 / 145	97.35

Table 4. The framework results of scenario 3.

From Tables 4 and 5, it can be verified that the larger the sample volume, the higher the accuracy rate obtained. In these scenarios, a greater traffic flow was presented in the five categories, which allowed obtaining more solid results in terms of the performance of the framework in relation to the first two scenarios. In this context, the average accuracy obtained was 95.58% and 93.79% for scenarios 2 and 4, respectively.

Scenario 4 presents a greater number of regions and movements both in a straight line and interceptions that increase its level of difficulty. In this scenario, the truck class was the class with the lowest performance, especially in the counting between the R3-R5 and R3-R4 regions. It was probably caused by large vehicles such as the truck or bus categories being counted in two output regions. This was because from the camera perspective it was not possible to obtain a significant separation between the two output regions.

From the results for the four scenarios, the general accuracy on the highways with straight-line movements was high (scenarios 1 and 3), while the accuracy on the highways with intercepts in T-conjunction and T-disjunction was lower (scenarios 2 and 4). Interception performance was affected by camera angle, which did not allow detection of some of the vehicles due to occlusions. In contrast, straight-line highways present a better perspective of vehicles, which minimizes occlusion problems and model detection errors.

Proper selection of the input and output regions prevents the same vehicle from being incorrectly counted in other regions. However, this is not a simple task, as in some scenarios the structures of the highway prevent an adequate separation between the regions from being achieved. An alternative to solve this problem, would be to focus the camera on specific regions of the highway and not the entire scene. The results of the experiments show that, despite the variation in the vehicle flow density and the errors caused by the occlusion, the framework does not decrease its counting performance. Additionally,

Category	R6 - R1 (CA / GT)	R2 - R5 (CA / GT)	R3 - R5 (CA / GT)	R3 - R4 (CA / GT)	Accuracy (%)
Car	2322 / 2398	2360 / 2389	328 / 329	105 / 111	97.4
Bus	33 / 38	11 / 11	23 / 24	0 / 0	94.2
Truck	131 / 137	138 / 153	23 / 19	7 / 6	86.98
Van	98 / 106	124 / 126	15 / 14	4 / 4	95.9
motorbike	296 / 294	279 / 285	50 / 54	15 / 17	94.45

Table 5. The framework results of scenario 4.

the framework is able to correct problems associated with detection failures that allow it to maintain the counting accuracy.

4.3 Running time results of the framework

In this session, the real-time rate was used to evaluate the speed of the proposed framework. The real time rate can be obtained by Equation 9, which is defined as the ratio between the time required for the framework to process a video and the playing time of the original video. When the value of the real-time rate is less than or equal to 1, the framework can perform real-time processing. On the other hand, the higher this value, the lower the capacity of the real-time processing framework.

$$real\ time\ rate = \frac{framework\ running\ time}{video\ running\ time} \tag{9}$$

The videos were cut to a duration of 50 minutes to better compare the results. This value was defined in relation to the shortest duration of one of the test videos. Table 6 shows the runtime of framework in the four test videos.

Scenario	Total vehicles	Video duration	Framework duration	Real time rate
1	1212	50 min	67.2 min	1.3
2	1416	50 min	71.4 min	1.4
3	3068	50 min	75.1 min	1.5
4	4600	50 min	97.1 min	1.9

Table 6. Runtime result in test videos.

From the table, it can be concluded that the proposed algorithm executed on the test equipment does not achieve real-time processing, since in all scenarios it obtained a real-time rate greater than 1.3.

Additionally, from the experiments performed, it can be verified that the number of vehicles processed by the framework directly affects the execution speed. As the number of vehicles in the video increases, the running time of the proposed framework also increases. This is because the algorithm has to identify more objects in each frame, which is equivalent to a greater number of operations performed internally. The complexity of the scenarios also affects the algorithm speed, as is the case of scenario 4, in which not only is there a greater vehicular flow, but also a greater number of regions to be evaluated, being the scenario with the highest execution times.

5. CONCLUSIONS AND FUTURE WORKS

In this work, a framework for detection and counting of vehicles in real traffic scenes was presented. To select the detection model that was used in the framework, seven one-stage detection models were compared in order to determine the model with the best trade-off between accuracy and speed in a new dataset. YOLOv4 achieved the best trade-off with mAP= 87.0% and a processing speed of 18 FPS.

The results with the framework show that the average count accuracy using YOLOv4 in the four test scenarios was 94.12%. One of the factors that affected the performance of the count by regions algorithm was the occlusions of smaller vehicles in both the input and output regions. One way to improve this problem would be to evaluate other camera positions, heights and/or angles that allow a better perspective of the highways and vehicles. The framework achieves a real-time processing rate of less than 1.9. This means that real-time processing is not possible with the equipment used in this research. However, more tests are necessary to determine its performance in equipment with greater processing capacity, and to be able to establish whether or not its implementation in a real vehicle traffic monitoring system is feasible.

ACKNOWLEDGEMENTS

The authors would like to thank PGEAS/UFSC for support the research project. This study was financed in part by the National Council for Scientific and Technological Development – Brazil (CNPq).

REFERENCES

Asha, C., Narasimhadhan, A., 2018. Vehicle counting for traffic management system using yolo and correlation filter. *2018 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, IEEE, 1–6. https://doi.org/10.1109/CONECCT.2018.8482380.

Bochkovskiy, A., Wang, C.-Y., Liao, H.-Y. M., 2020. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv preprint arXiv:2004.10934*.

Chen, Q., Huang, N., Zhou, J., Tan, Z., 2018. An ssd algorithm based on vehicle counting method. *2018 37th Chinese Control Conference (CCC)*, IEEE, 7673–7677. https://doi.org/10.23919/ChiCC.2018.8483037.

Ciampi, L., Amato, G., Falchi, F., Gennaro, C., Rabitti, F., 2018. Counting vehicles with cameras. *SEBD*.

Dai, Z., Song, H., Wang, X., Fang, Y., Yun, X., Zhang, Z., Li, H., 2019. Video-based vehicle counting framework. *IEEE Access*, 7, 64460–64470. https://doi.org/10.1109/ACCESS.2019.2914254.

Dalal, N., Triggs, B., 2005. Histograms of oriented gradients for human detection. *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, 1, Ieee, 886–893. https://doi.org/10.1109/CVPR.2005.177.

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., Zisserman, A., 2010. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2), 303–338. https://doi.org/10.1007/s11263-009-0275-4.

Girshick, R., 2015. Fast r-cnn. *2015 IEEE International Conference on Computer Vision (ICCV)*, 1440–1448. https://doi.org/10.1109/ICCV.2015.169.

Girshick, R., Donahue, J., Darrell, T., Malik, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 580–587.

Hardjono, B., Rhizma, M. G., Widjaja, A. E., Tjahyadi, H., Josodipuro, M. J., 2019. Vehicle counting evaluation on low-resolution images using software tools. *Proceedings of the 9th International Conference on Information Communication and Management*, 89–94. https://doi.org/10.1145/3357419.3357453.

He, K., Gkioxari, G., Dollár, P., Girshick, R., 2017. Mask r-cnn. *Proceedings of the IEEE international conference on computer vision*, 2961–2969.

He, K., Zhang, X., Ren, S., Sun, J., 2015. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9), 1904–1916.

Juan, L., Gwun, O., 2009. A comparison of sift, pca-sift and surf. *International Journal of Image Processing (IJIP)*, 3(4), 143–152.

Lin, J.-P., Sun, M.-T., 2018. A yolo-based traffic counting system. *2018 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, IEEE, 82–85. https://doi.org/10.1109/TAAI.2018.00027.

Lin, T., Goyal, P., Girshick, R., He, K., Dollár, P., 2017. Focal loss for dense object detection. *2017 IEEE International Conference on Computer Vision (ICCV)*, 2999–3007. https://doi.org/10.1109/ICCV.2017.324.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C. L., 2014. Microsoft coco: Common objects in context. *European conference on computer vision*, Springer, 740–755. https://doi.org/10.1007/978-3-319-10602-1_48.

Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., Pietikäinen, M., 2020. Deep learning for generic object detection: A survey. *International journal of computer vision*, 128(2), 261–318. https://doi.org/10.1007/s11263-019-01247-4.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., Berg, A. C., 2016a. Ssd: Single shot multibox detector. *European conference on computer vision*, Springer, 21–37. https://doi.org/10.1007/978-3-319-46448-0_2.

Liu, X., Liu, W., Ma, H., Fu, H., 2016b. Large-scale vehicle re-identification in urban surveillance videos. *2016 IEEE International Conference on Multimedia and Expo (ICME)*, IEEE, 1–6. https://doi.org/10.1109/ICME.2016.7553002.

Mandal, V., Adu-Gyamfi, Y., 2020. Object Detection and Tracking Algorithms for Vehicle Counting: A Comparative Analysis. *Journal of Big Data Analytics in Transportation*, 1–11. https://doi.org/10.1007/s42421-020-00025-w.

Mita, T., Kaneko, T., Hori, O., 2005. Joint haar-like features for face detection. *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, 2, IEEE, 1619–1626.

Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016. You only look once: Unified, real-time object detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779–788. https://doi.org/10.1109/CVPR.2016.91.

Redmon, J., Farhadi, A., 2017. Yolo9000: Better, faster, stronger. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6517–6525. https://doi.org/10.1109/CVPR.2017.690.

Redmon, J., Farhadi, A., 2018. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767.*

Ren, S., He, K., Girshick, R., Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 91–99.

Rothe, R., Guillaumin, M., Van Gool, L., 2014. Non-maximum suppression for object detection by passing messages between windows. *Asian conference on computer vision*, Springer, 290–306. https://doi.org/10.1007/978-3-319-16865-4_19.

Santos, A. M., Bastos-Filho, C. J., Maciel, A. M., Lima, E., 2020. Counting vehicle with high-precision in brazilian roads using yolov3 and deep sort. *2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, IEEE, 69–76. https://doi.org/10.1109/SIBGRAPI51738.2020.00018.

Song, H., Liang, H., Li, H., Dai, Z., Yun, X., 2019. Vision-based vehicle detection and counting system using deep learning in highway scenes. *European Transport Research Review*, 11(1), 1–16. https://doi.org/10.1186/s12544-019-0390-4.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015. Going deeper with convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1–9.

Tan, M., Pang, R., Le, Q. V., 2020. Efficientdet: Scalable and efficient object detection. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10781–10790.

Wojke, N., Bewley, A., 2018. Deep cosine metric learning for person re-identification. *2018 IEEE winter conference on applications of computer vision (WACV)*, IEEE, 748–756. https://doi.org/10.1109/WACV.2018.00087.

Yosinski, J., Clune, J., Bengio, Y., Lipson, H., 2014. How transferable are features in deep neural networks? *arXiv preprint arXiv:1411.1792.*

Zhang, J., Wang, F.-Y., Wang, K., Lin, W.-H., Xu, X., Chen, C., 2011. Data-driven intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 12(4), 1624–1639. https://doi.org/10.1109/TITS.2011.2158001.

Zhang, S., Wu, G., Costeira, J. P., Moura, J. M., 2017. Fcn-rlstm: Deep spatio-temporal neural networks for vehicle counting in city cameras. *Proceedings of the IEEE international conference on computer vision*, 3667–3676.

Zhao, Z.-Q., Zheng, P., Xu, S.-t., Wu, X., 2019. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11), 3212–3232. https://doi.org/10.1109/TNNLS.2018.2876865.

Zhou, X., Wang, D., Krähenbühl, P., 2019. Objects as points. *arXiv preprint arXiv:1904.07850.*