applied sciences

MDPI

Article

Efficient Roundabout Supervision: Real-Time Vehicle Detection and Tracking on Nvidia Jetson Nano

Imane Elmanaa [1,2,*], My Abdelouahed Sabri [1,*], Yassine Abouch [2] and Abdellah Aarab [1]

[1] LISAC Laboratory, Faculty of Sciences Dhar El Mahraz, Sidi Mohamed Ben Abdellah University, Fes 30000, Morocco
[2] DAKAI Laboratory, Nextronic by Aba Technology, Casablanca 20253, Morocco
* Correspondence: imane.elmanaa@usmba.ac.ma (I.E.); abdelouahed.sabri@gmail.com (M.A.S.); Tel.: +212-634781637 (I.E.); +212-600504321 (M.A.S.)

Abstract: In recent years, a significant number of people in Morocco have been commuting daily to Casablanca, the country's economic capital. This heavy traffic flow has led to congestion and accidents during certain times of the day as the city's roads cannot handle the high volume of vehicles passing through. To address this issue, it is essential to expand the infrastructure based on accurate traffic-flow data. In collaboration with the municipality of Bouskoura, a neighboring city of Casablanca, we proposed installing a smart camera on the primary route connecting the two cities. This camera would enable us to gather accurate statistics on the number and types of vehicles crossing the road, which can be used to adapt and redesign the existing infrastructure. We implemented our system using the YOLOv7-tiny object detection model to detect and classify the various types of vehicles (such as trucks, cars, motorcycles, and buses) crossing the main road. Additionally, we used the Deep SORT tracking method to track each vehicle appearing on the camera and to provide the total number of each class for each lane, as well as the number of vehicles passing from one lane to another. Furthermore, we deployed our solution on an embedded system, specifically the Nvidia Jetson Nano. This allowed us to create a compact and efficient system that is capable of a real-time processing of camera images, making it suitable for deployment in various scenarios where limited resources are required. Deploying our solution on the Nvidia Jetson Nano showed promising results, and we believe that this approach could be applied in similar traffic-surveillance projects to provide accurate and reliable data for better decision-making.

Keywords: smart camera; YOLOv7-tiny; object detection; Deep SORT; tracking; embedded system; Nvidia Jetson Nano; vehicle counting

Citation: Elmanaa, I.; Sabri, M.A.; Abouch, Y.; Aarab, A. Efficient Roundabout Supervision: Real-Time Vehicle Detection and Tracking on Nvidia Jetson Nano. *Appl. Sci.* **2023**, *13*, 7416. https://doi.org/10.3390/app13137416

Academic Editors: Junchi Yan and Minghao Guo

Received: 6 April 2023
Revised: 25 May 2023
Accepted: 6 June 2023
Published: 22 June 2023

1. Introduction

Bouskoura is a Moroccan city located approximately twenty kilometers south of the economic capital, Casablanca. Recently, the city has experienced a significant increase in population, resulting in a rise in road traffic as many individuals commute to and from Casablanca for work. This has led to direct congestion, particularly at intersections, resulting in increased travel time, vehicle blockages, and heightened fuel consumption. Congestion can cause driver stress and frustration [1], which can significantly impact individuals' performance and efficiency, lowering economic productivity and reducing the quality of life. Therefore, it is crucial to develop a traffic-management strategy that improves the design of existing roads [2].

Deep learning is a rapidly growing subfield of artificial intelligence that has shown a remarkable ability to learn from vast amounts of data and can solve complex problems with high accuracy [3,4]. However, this often requires access to powerful computing resources and significant data storage, making it challenging to deploy deep-learning algorithms in resource-limited environments [5,6].

To address this challenge, our study proposes a solution that leverages deep-learning techniques to analyze the traffic data collected by our smart camera and to make informed decisions on which road lanes to expand or extend. By designing our solution with consideration for resource constraints, we can demonstrate the feasibility of deploying deep learning in real-world applications with limited resources. Deep learning has already shown great potential for improving traffic management and our solution provides a promising avenue for addressing traffic congestion challenges in urban environments.

In this paper, we present an efficient system that leverages modern image processing and artificial-intelligence technologies to support decision-making regarding which road lanes to extend or expand. Our system uses a smart camera equipped with an Nvidia Jetson Nano card to gather statistics. The camera records the number of vehicles by category at each intersection lane daily and calculates the precise count of vehicles moving from a certain entrance to a certain exit. All processing is carried out on the card, and the resulting statistics are then transmitted via the standard IoT messaging protocol MQTT [7] for further analysis and decision support.

Our initial task was to create an application that could enable us to gather a substantial dataset to construct our detection and classification model. For data collection, we utilized an Allwinner A133 embedded card with a Linux system. The card was programmed to capture video sequences from an IP camera installed at the same intersection. The data were then stored in an external memory card and transmitted to a server via FTP with a secondary program.

We used these videos to create a dataset of 13,000 images. For object detection, we employed YOLOv7 (you only look once, version 7), which is currently the most reliable and easily deployable model available [8]; it was selected based on these qualities, as well as its compatibility with the Nvidia Jetson Nano card. The images were manually labeled into four classes: cars, trucks, motorcycles, and buses. Manual labeling was essential to ensure the accuracy and precision of the labeled data, resulting in a higher-quality classification model. By manually labeling the images, we were able to carefully inspect and classify each object in the images, reducing the likelihood of errors or misclassifications. Our model of vehicle detection and classification was trained using 7000 images for training, 4000 for validation, and 2000 for testing. We evaluated the performance of our trained model using several measures, including a mean average precision (mAP) at a threshold of 0.5 of 91%, a recall of 84.1%, and a precision of 87.2%. After detection and recognition, we utilized the Deep SORT algorithm for vehicle tracking. Finally, the counting of vehicles on each road at the intersection was conducted by using a mathematical formula that calculated the intersection of two segments to trigger a vehicle count in increments. Finally, we calculated the number of vehicles passing from one road to another; this calculation was based on tracking that allows us to identify each vehicle and follow it throughout the scene with the information found previously, which is the number of vehicles passing through each road.

Our main contribution is the development of a robust and efficient vehicle-counting solution that can be deployed on the Nvidia Jetson Nano card. We achieved this by leveraging a large dataset, which was collected with an attention to different scenarios and conditions. This allowed us to train a high-performing detection model that can detect vehicles with high accuracy and low latency. In addition to detection, we utilized a tracking algorithm, Deep SORT, which maintains the identity of each vehicle as it moves throughout the roundabout, improving the counting accuracy of the system. Our solution offers several advantages over traditional methods, including speed, accuracy, and scalability. Deploying the solution on the Nvidia Jetson Nano card ensures low latency and high performance, making it ideal for real-time applications. In our experimental results, we demonstrated that our solution outperforms existing methods, achieving high accuracy and low error rates. We believe that our contribution can pave the way for more efficient and accurate vehicle-counting systems, which can have significant applications in areas such as traffic management, security, and surveillance.

The remainder of this paper is structured as follows: Section 2 presents the previous works that deal with similar projects, i.e., "counting vehicles in embedded systems". Section 3 provides a brief overview of our project, including the cost of the solution and a detailed description of the overall architecture. Each component of the architecture is discussed in a separate subsection. Section 4 explains the experimental results. Section 5 includes a discussion and, finally, a conclusion.

2. Related Work

The main objective of our paper is to develop an intelligent camera system capable of processing a camera feed in real time to calculate the number of vehicles in each category passing through an intersection, as well as the total number of vehicles passing through each path. In the literature, we found various pieces of research and projects related to this problem and we would like to cite some examples.

Abuelgasim et al. [9] presented a vehicle-counting system that utilized the YOLOv5 algorithm for object detection and Deep SORT for tracking. They collected a database of vehicles with four classes and deployed their system on an Nvidia Jetson Nano card, which can process at a speed of 15 fps. However, their system was only tested on videos; to truly evaluate the performance of a solution, it should be tested in real-time conditions under various lighting and weather conditions. Therefore, to ensure optimal performance, it is necessary to train a robust model that can handle different lighting and weather conditions.

Duc-Liem et al., 2021 [10] presented a low-cost and effective system that integrated object detection models to detect, track, and count vehicles. They created a vehicle detection dataset representing traffic conditions in Vietnam and evaluated several deep learning models on two different types, a Coral Dev Board that supported TPU and then an Nvidia Jetson Nano Board that supported GPU. However, their study acknowledged the need to improve the inference time of the model for optimal performance. To address this limitation, our work aims to optimize the inference time, thus making a system that is even more practical for traffic control and management.

Valladares et al., 2021 [11] assessed the performance of Nvidia cards by using machine learning to count vehicles in real time in the city of Quito, Ecuador. They evaluated performance based on several factors, including resource utilization (GPU and CPU), power consumption, the temperature of the card, and the RAM usage during processing. Additionally, they confirmed the need to increase the precision of their model, as the use of OpenDataCam for vehicle-counting and identification proved effective, but certain factors such as camera position, lighting, vehicle size and speed, and traffic flow needed to be considered. Therefore, it is necessary to find a solution that can adapt to all these conditions.

Yan Han et al., 2017 [12] developed a driving-assistance system utilizing embedded cards, such as the Nvidia Jetson. The system consists of two components: the first one utilizes image processing techniques to locate signs, while the second uses a convolutional neural network (CNN) to classify the detected signs. The proposed system was implemented on the Nvidia Jetson TX1 board with a web camera. Although the system achieved a high detection accuracy of 96%, the frame rate was low, at 1.6 frames per second (FPS). Therefore, a faster model is needed to be deployed on the Jetson card.

Zhihui Wang et al., 2020 [13], proposed an integrated solution that combines detection, tracking, and trajectory modeling algorithms for vehicle detection and counting. They utilized a GMM-type backward modeling approach along with a machine-learning-based detector to enhance the accuracy of vehicle detection and counting precision. They also implemented a trajectory modeling scheme that considers the direction and trajectory of the vehicles. However, their algorithm still has limitations, such as typical detection failures, false positives due to dynamic backgrounds or reflections, and missing detections of vehicles that are too large or too small. Therefore, there is a need to consider more robust algorithms to overcome these issues.

We have taken into consideration the limitations and difficulties highlighted by previous studies in the field of vehicle detection and counting. To improve the performance

of this research, we have proposed the use of alternative algorithms for object detection and a new method of vehicle-counting that is adaptable to the specific requirements of the system. Additionally, we have collected a new database of vehicles that considers all the possible detection difficulties, such as occlusion, illumination variations, or bad weather conditions, to ensure a more robust and reliable detection and counting system.

3. System Architecture and Test Environment

The proposed traffic-management system aims to analyze the traffic at an intersection located between Bouskoura and Casablanca. For this purpose, we installed an IP camera connected to an Nvidia Nano Jetson card, which serves as our processing unit. Figure 1 shows some of the images captured during the installation. Our system functions as a smart camera capable of detecting and tracking vehicles in real time. One of the most important features of our camera is that it follows the concept of edge computing, which means that the analysis is performed directly on the device, and only the counting results are transmitted. This has two main advantages: first, we do not need to transmit the images, which reduces bandwidth requirements; second, we respect the privacy of citizens, which is a crucial consideration for any application deployed in a public place.

Figure 1. Installation images of our smart vehicle counter system.

To provide a practical and informative solution, we have included the current cost of each piece of equipment used in our system. Table 1 summarizes the estimated cost of each component:

Table 1. Cost breakdown of equipment used in the solution.

Component	Model	Cost
IP camera	Hikvision Varifocal Motorized Bullet Camera	USD 270
AI Processing Unit	Nvidia Jetson Nano 4 GB Development Kit	USD 214.63
Accessories (cabling, protection, etc.)		Approximately USD 200

The system architecture of the proposed traffic management system is composed of several components, including data acquisition, object detection and classification, object tracking and identification, counting, and data transmission. The data-acquisition component captures video footage of the traffic at each intersection using a smart camera and statistics are calculated and transmitted via the MQTT protocol. The object detection and classification component uses the YOLOv7-tiny algorithm for vehicle identification and classification. The object tracking and identification component uses the Deep SORT algorithm to track and identify vehicles. The counting component combines image processing techniques and deep-learning algorithms to count the number of vehicles passing through

each lane. Finally, the data-transmission component is responsible for transmitting the processed traffic data to a centralized server for further analysis and decision-making via the MQTT protocol.

Figure 2 shows the architectural overview of the proposed system, including its components.

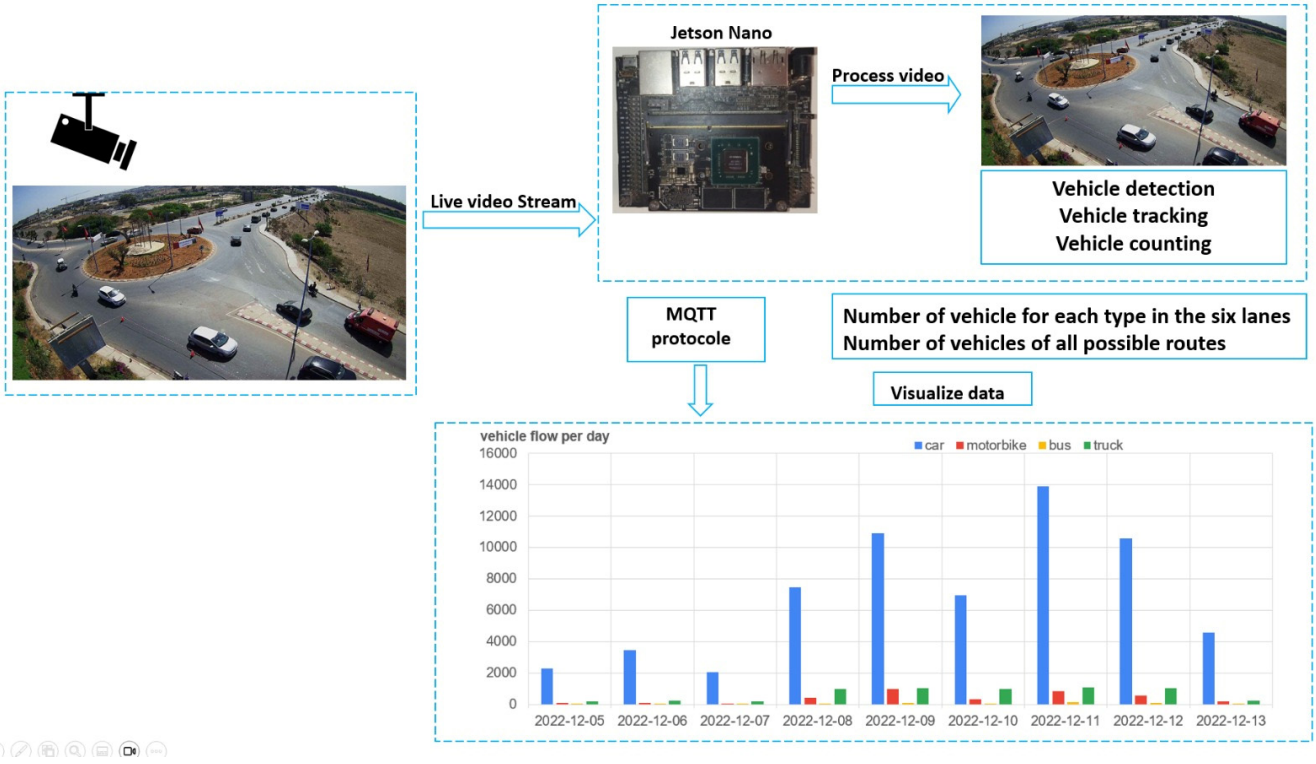


Figure 2. The proposed architecture for traffic flow detection. Vehicle detecting, tracking, and counting are directly performed in the Nvidia Jetson Nano.

3.1. Collection and Description of Data

Before deciding to build a custom detection model, we tested several existing models based on other datasets, such as COCO. However, these models did not meet the desired level of accuracy due to the constraint of having only one camera covering all the lanes of the intersection. The camera captures large vehicles that are close but the problem arises with vehicles in other lanes that are further away and appear smaller in size. To address this issue, we decided to build a dataset that met our needs by using the same camera deployed in the intersection. This approach ensured that the detection and tracking were optimized for the specific camera and consequently improved the accuracy of the counting, which is the primary purpose of our application.

To build our dataset, we used an embedded system with a Linux system based on the Allwinner A133 motherboard. We programmed the board to capture video footage throughout the day from the IP camera installed at the intersection. The videos were stored on a memory card and sent to the server via FTP with a Python program that removed the successfully transmitted videos.

We carefully filtered the collected video sequences to obtain images that contained diversified sets representing different classes (cars, trucks, buses, and motorcycles) at different times of the day and in varying weather conditions, ensuring that all possible variations of the real world were included.

The dataset contains 13,000 images, which were manually annotated using the labeling tool with the text extension of YOLO.

Figure 3 shows the system architecture used for the dataset collection.

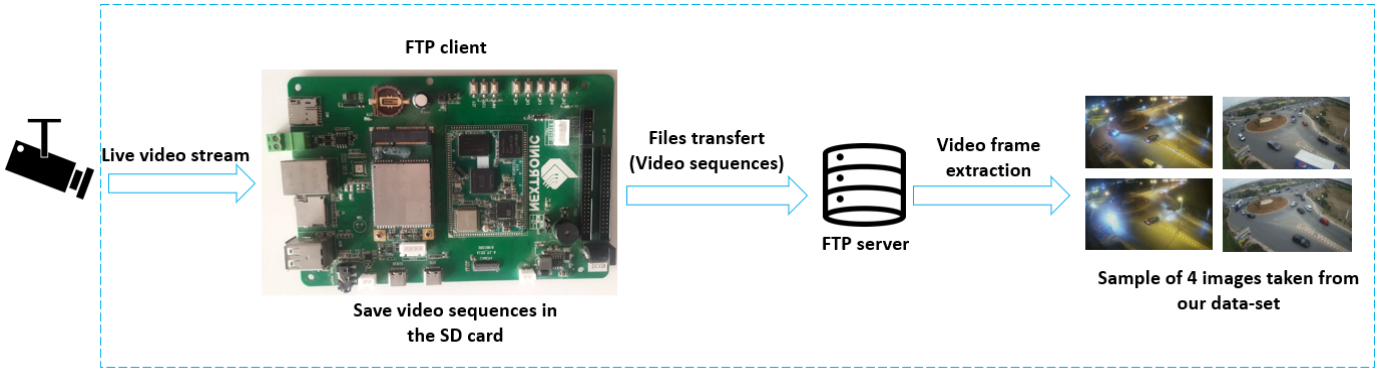


Figure 3. The architecture of the system used for the dataset collection.

3.2. Object Detection, YOLOv7

To select the best algorithm for our vehicle detection and classification task, we needed to consider several criteria, including real-time detection with high reliability and deployment ability on the Nvidia Jetson Nano card. After careful consideration, we chose the YOLOv7 as the most suitable detection algorithm for our application as it has been successfully tested in several real-world applications [8,14,15]. YOLOv7 is characterized by a unique CNN architecture that processes multiple networks separately [8].

In a YOLO model (the general structure [8] of which is represented in Figure 4), the back part is the part that groups the pixels of the image. The representations of the layers of the convolutional network are combined and blended in the neck; then, they are passed to the prediction head. The head is the final part of the CNN, in which our YOLO model predicts the object classes and locations.

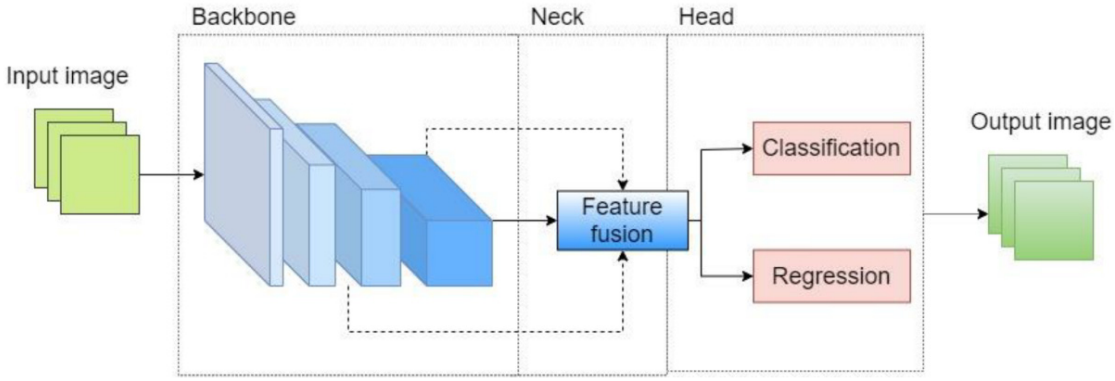


Figure 4. YOLO network architecture.

We carried out a comparative analysis of YOLOv7 with other YOLO versions, including YOLOv4 and YOLOv5. Our study revealed that YOLOv7 outperforms both YOLOv4 and YOLOv5 in terms of accuracy, parameters, and computation. Specifically, YOLOv7 showed a 1.5% higher average precision (AP) than YOLOv4-tiny, while reducing the number of parameters by 75% and computation by 36%. Moreover, the more edge-optimized version of YOLOv7, YOLOv7-tiny, achieved the same AP as YOLOv4-tiny, while using 39% fewer parameters and requiring 49% less computation. When compared to YOLOv5-N, YOLOv7-tiny demonstrated a 10.7% higher accuracy on AP and was 127 FPS faster [16].

Our custom model for vehicle detection and classification was trained using the transfer learning approach, and it was based on the pretrained weight file of YOLOv7-tiny [17] and our annotated dataset. We ran YOLOv7-tiny on the Nvidia Jetson Nano as this model can work faster on this type of device. Overall, our comparative study and choice of YOLOv7 demonstrate that it is a highly effective and efficient algorithm for real-time vehicle detection and classification.

3.3. Vehicle Tracking

In object tracking applications, there is the problem of correctly identifying the same object over multiple frames of a video. Deep SORT addresses this issue by assigning a unique identification (ID) to each detected object and by using a combination of appearance and motion information to track the object over time. To achieve the objective of our project, which is the counting of vehicles, we need to track all the vehicles accurately, which is a crucial task in this type of application. The tracking of objects involves detecting the object, identifying it in a frame, and tracking it across all sequences until it leaves the scene. To achieve this task, we chose to use the Deep SORT algorithm (simple online and real-time tracking with a deep association metric), which is a multiobject tracking method. The Deep SORT algorithm uses the spatial and temporal characteristics of the targets to track and maintain the identification of all moving objects in all sequences [18] (as shown in Figure 5). By assigning each detected object a unique ID, Deep SORT ensures that the same vehicle is consistently identified and tracked across multiple frames, allowing for accurate vehicle-counting.

Figure 5. Deep SORT algorithm result, a unique ID is given to each class of detected vehicle.

The Deep SORT algorithm consists of the following steps (as is shown in Figure 6): First, the vehicles are detected using the YOLOv7-tiny. Then comes the multiobject tracking step, where the Kalman filter predicts the trajectories based on the position and speed of the target object; meanwhile, the Hungarian algorithm measures the correlation [19].

Deep SORT distinguishes itself by using a combination of two convolutional layers, followed by six residual blocks and an appearance descriptor [19]. This makes the system more robust against missing-object occlusions by significantly reducing the number of identity changes. This combination makes Deep SORT an appropriate choice for applications requiring real-time monitoring. Furthermore, Deep SORT has been shown to be effective in handling complex scenarios with significant occlusion, which is a common challenge in tracking vehicles that are using a roundabout. The algorithm is also capable of tracking vehicles with varying speeds and trajectories, making it ideal for real-world applications.

Additionally, the ability of Deep SORT to maintain the identity of each vehicle as it moves throughout the roundabout helps to improve the accuracy of the vehicle-counting system. Overall, the combination of traditional tracking methods with deep-learning techniques, the ability to handle complex scenarios, and the ability to maintain the identity of each vehicle makes Deep SORT a suitable choice for vehicle tracking for vehicles

using a roundabout, and it contributes to the high accuracy of the proposed vehicle-counting solution.

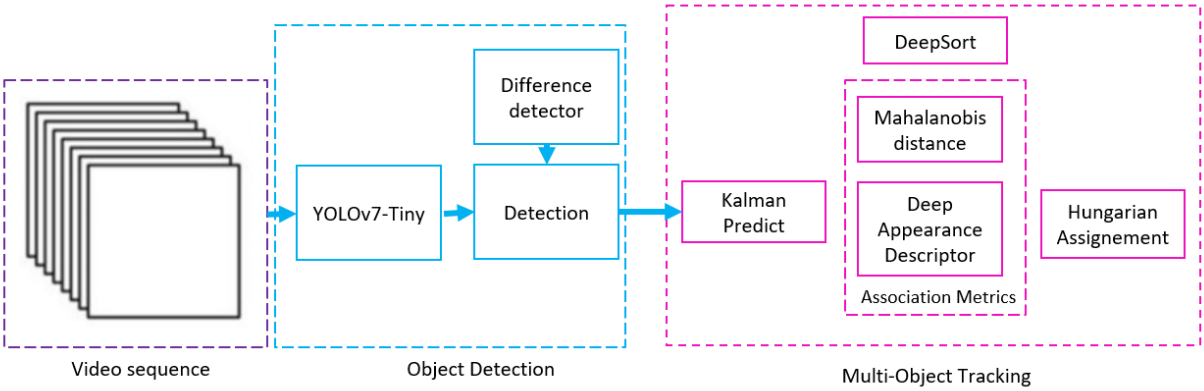


Figure 6. The architecture of the Deep SORT algorithm.

3.4. The Counting of Vehicles at Intersections

The principle used for the counting of vehicles in the different lanes [20] of the intersection involves drawing virtual custom lines that cut across the six lanes of the roundabout. Figure 7 shows an illustrative diagram of the method used. Each line drawn must reach both sides of each lane in a single direction. For the detecting and tracking of vehicles in different images, the system calculates the intersection between the defined line and the segment composed of two points: the centroid of the current box of the vehicle and the centroid of the old box of the same detected vehicle (Figure 8). When the segment crosses the imaginary line, the system calculates the number of vehicles in increments and detects the vehicles in the relevant category.

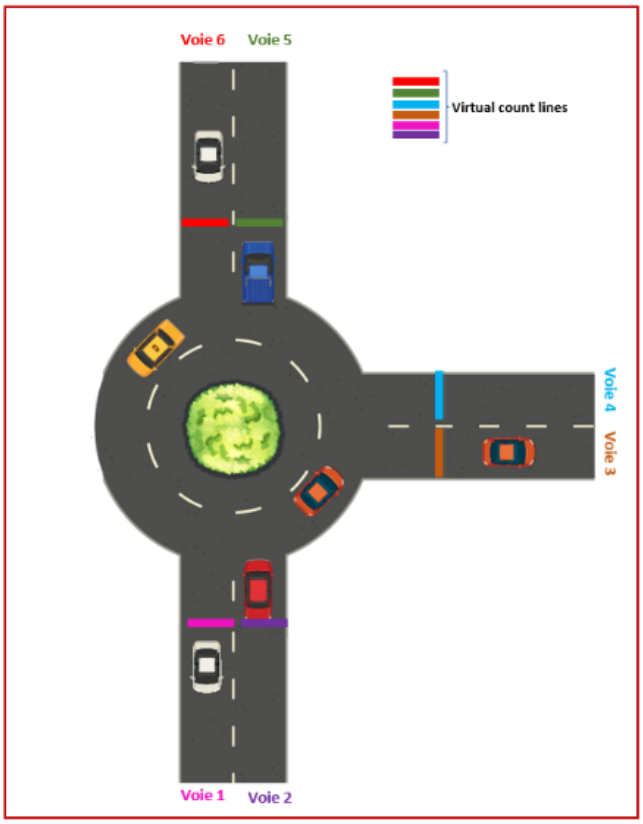


Figure 7. Illustrative diagram of the virtual lines used for counting the vehicles.

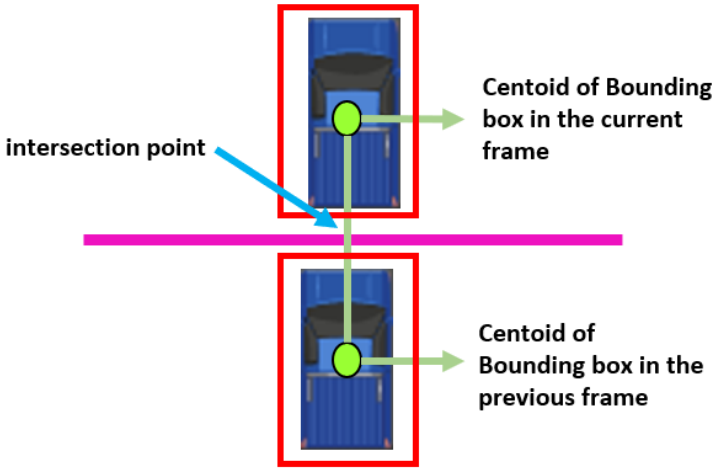


Figure 8. The mechanism used to check if a bounding box was intersected by the count line.

To calculate the number of vehicles passing from one lane to another, we recorded the IDs of the vehicles that crossed each lane. If a certain vehicle with the same ID crossed two lanes simultaneously, we calculated the number of vehicles, in increments, that were making this maneuver. For example, if the vehicle identified by the number 1 passes through lane 3 at first, and then passes through lane 4, we calculated the number, in increments, of vehicles that passed from lane 3 to lane 4.

In Figure 9, we give an example of the possible trajectories for a vehicle coming from lane 2. As illustrated in the figure, we can conclude that a vehicle coming from lane 2 can pass through lane 5, lane 3, or lane 1, thus making the following routes V2 $\rightarrow$ V5, V2 $\rightarrow$ V3, and V2 $\rightarrow$ V1. Additionally, the same principle applies to all the other lanes of the intersection.

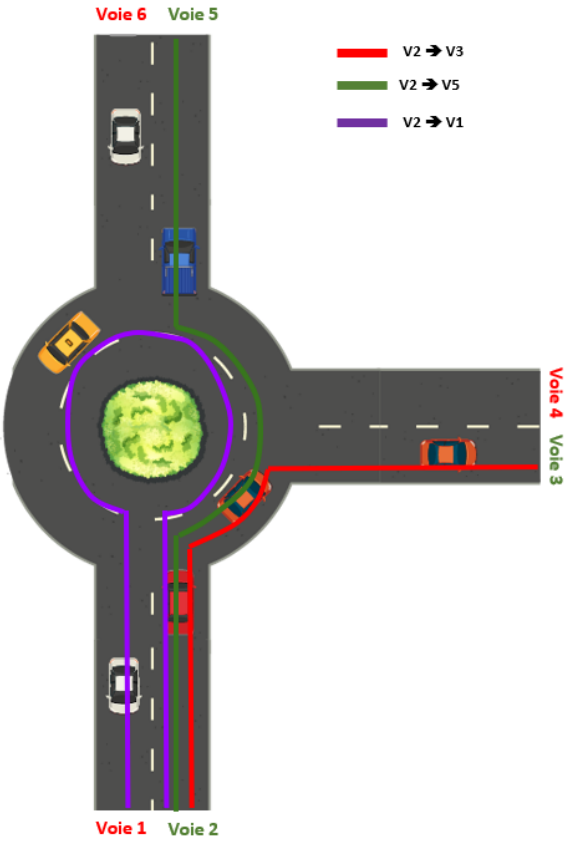


Figure 9. Diagram that illustrates the possible trajectory of vehicles coming from Lane 2.

3.5. Deploy Model for the Nvidia Jetson Nano

The Nvidia Jetson Nano is a small, low-power computer designed for use in embedded systems and other applications requiring a high computing performance within a compact form factor. It is based on the Nvidia Jetson platform and is powered by a quad-core ARM Cortex-A57 CPU and a 128-core Maxwell GPU. It supports a wide range of interfaces and peripherals, including USB, ethernet, and HDMI [21]. The Nvidia Jetson Nano has a maximum power consumption of 10 watts, which makes it an energy-efficient solution for edge computing. It also has a memory bandwidth of 25.6 GB/s, which allows for fast and efficient data transfer between the CPU and GPU. These parameters make the Nvidia Jetson Nano a suitable platform for running deep learning models as it can handle complex computations while maintaining a low power consumption. Additionally, the Nvidia Jetson Nano is equipped with 4 GB of LPDDR4 memory, which provides sufficient memory for running deep learning models and handling large datasets. Overall, the Nvidia Jetson Nano's power consumption and memory bandwidth parameters make it an excellent choice for deploying AI solutions at the cutting edge.

4. Results

In this part, we present the performance evaluation of the deployed system, the performance of the vehicle-detection model trained by YOLOv7-tiny, the efficiency of the counting method used, and present the counting accuracy scores.

4.1. Training Results

Our model of vehicle detection and classification was trained using 7000 images for training, 4000 for validation, and 2000 for testing. We evaluated the performance of our trained model using several measures, a mAP@.5 of 91%, a recall of 84.1%, and a precision of 87.2%.

We trained our model using the architecture of the YOLOv7-tiny model as a starting point for the training process. The model was trained for 300 epochs, which took 8.715 h with the use of an Nvidia RTX 3060ti GPU computer. Figure 10 shows the plots of the functions of losses. It contains the loss of box regression that shows the way in which the boxes cover the vehicles, and also the capacity to locate the centers of each category of vehicles. In addition, it includes the classification loss, which represents the ability to differentiate between the different detected objects, and finally the abjectness loss, which is the probability that the detected vehicle exists in the ROI.

The training results of our pretrained YOLOv7-tiny model for object detection show that the precision, recall, and mAP increase as the number of epochs increases, indicating that the training process was effective. On the other hand, the errors in objectness, classification, and box decreased with the number of epochs, indicating that our model is well-trained.

The precision–recall curve is a graph that shows the trade-off between the precision and recall for a classifier. In the case of the YOLOv7-tiny model that was trained for vehicle detection and classification, the precision–recall curve is shown in Figure 11 and it is based on the precision and recall data obtained for each class (i.e., car, truck, bus, and motorcycle).

From the obtained results, we can calculate the average precision (mAP) for all classes at a threshold of 0.5. The average precision for all classes is 0.801, which indicates that the model can detect and classify vehicles with high precision.

4.2. Experimental Results

To evaluate our application that counts vehicles in an intersection in the real world, we used the Nvidia Jetson Nano as a peripheral computing system. The experimental device was deployed at the intersection between Bouskoura and Casablanca (see Figure 1). We used the Nvidia Jetson Nano to perform all the necessary steps in order to conduct the analysis, detection, tracking, and counting. The final step involved sending the counting results via MQTT to aid in the decision-making regarding expanding the intersection to mitigate congestion.

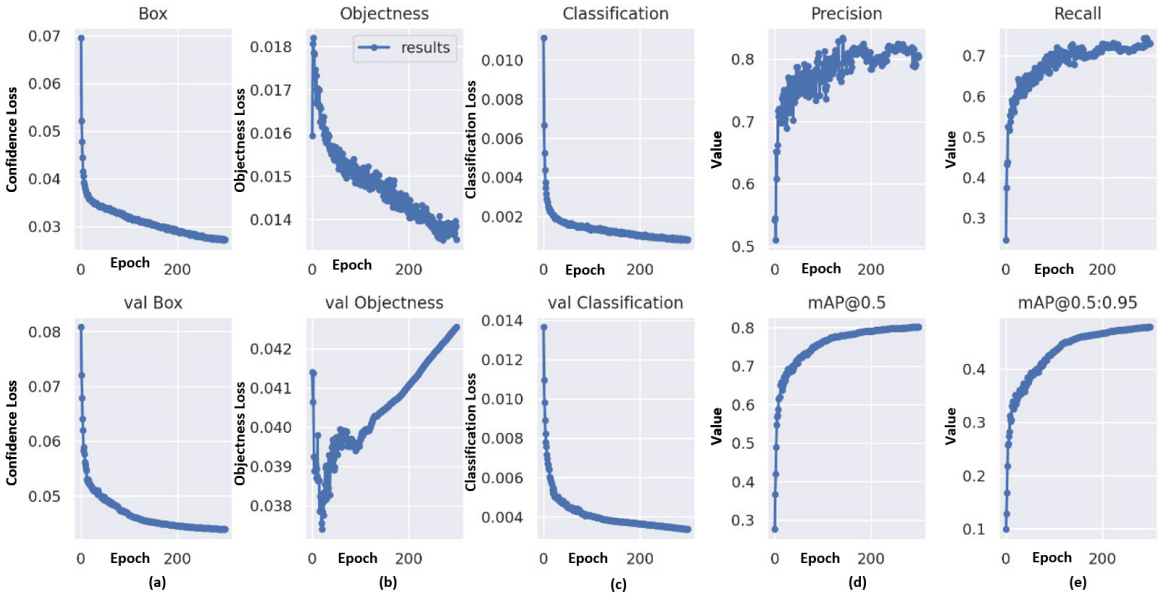


Figure 10. The training model results of the pretrained YOLOv7-tiny for object detection. (**a**) Coordinate Loss; (**b**) Objectness Loss; (**c**) Classification Loss; (**d**) Precision; and (**e**) Recall.

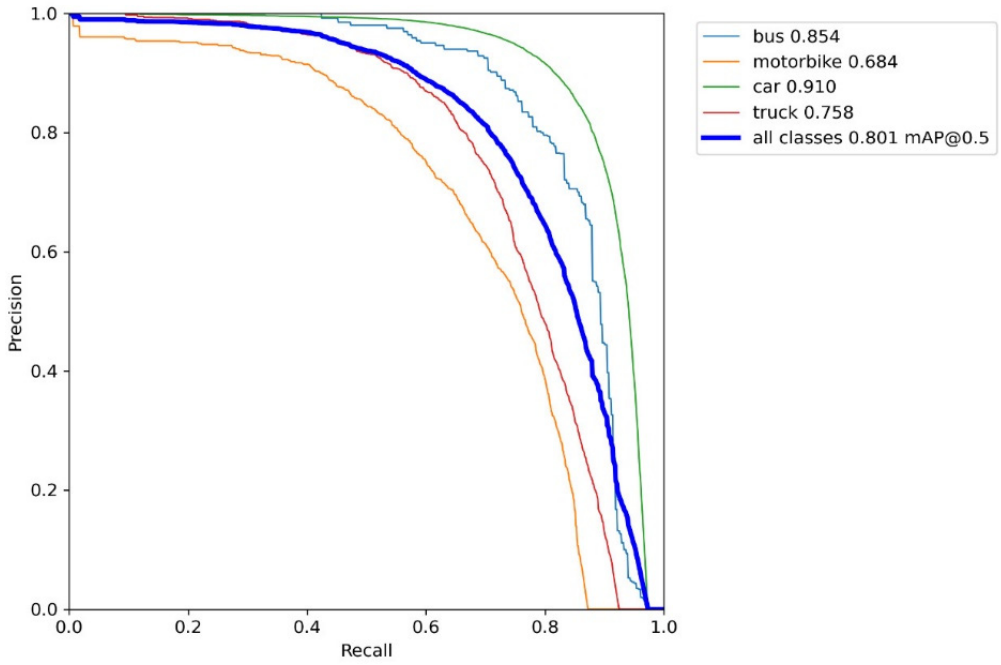


Figure 11. Precision–recall curve.

The detection, tracking, and counting results were displayed on a monitor connected via HDMI. We have demonstrated an example of the results obtained with an average inference execution speed of 16 FPS in Figure 12. In addition, we captured two photos, one taken during the day and the other taken at night, to showcase the performance of our algorithm in different lighting conditions. The first photo was captured on 24 October 2022, at 06:44:26 with low lighting conditions. The other was taken at 13:44:00 on the same day. These results demonstrate the effectiveness of our solution in detecting and tracking vehicles under various lighting conditions.

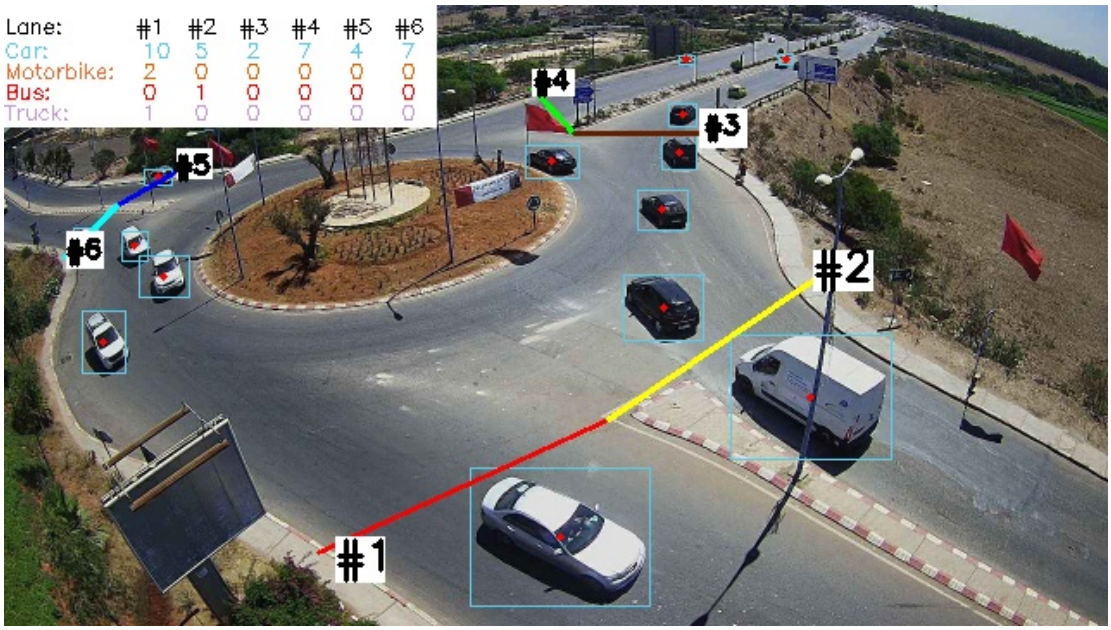


Figure 12. Example of a simulation result.

4.2.1. System Evaluation

To evaluate the performance of our counting system, we use four metrics: precision, recall, F-measure, and accuracy. These metrics were calculated using the following parameters: true positive (TP), which represents the successfully counted vehicles; false negative (FN), which represents the vehicles that our system is not able to count; and false positive (FP), which represents the vehicles that are incorrectly counted.

The mathematical relationships between these parameters and the evaluation metrics are shown below:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

$$\text{F-measure} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

$$\text{Accuracy} = 1 - \frac{\text{counted number} + \text{real number}}{\text{real number}} \tag{4}$$

Our system has shown promising results in most of the tested videos. The table (Figure 13) displays the counting results of a previously tested video, with each of the six lanes at the intersection represented in the table. However, we did observe instances of false negatives in lanes 4, 5, and 6, which may be due to the detection not being entirely reliable, possibly due to the significant distance between the camera and these lanes.

4.2.2. Data Visualization

From the results sent by our smart camera installed at the intersection, we drew the diagrams below. Figure 14 shows the counting results for the period from 5 December 2022 to 13 December 2022. It also shows the number of cars, trucks, buses, and motorcycles passing through each lane of the intersection. By analyzing these numbers, we can easily determine which lanes have the most traffic, as well as which days of the week have the most traffic. In addition, we specifically chose the days included in the diagrams because they occurred during rainy weather, and our model has been proven to be reliable in detecting vehicles in different weather conditions.

Lane	Vehicle	real number	counted number	TP	FN	FP	Precision	Recall	F-measure	Accuracy
1	Car	96	97	96	0	1	0.989	1	0.994	0.989
	Truck	12	12	12	0	0	1	1	1	1
	Motorbike	14	12	12	2	0	1	0.857	0.922	0.857
	Bus	10	10	10	0	0	1	1	1	1
2	Car	36	37	36	0	1	0.972	1	0.985	0.972
	Truck	10	9	9	1	0	1	0.9	0.947	0.9
	Motorbike	5	3	3	2	0	1	0.6	0.75	0.6
	Bus	12	12	12	0	0	1	1	1	1
3	Car	24	23	22	2	1	0.956	0.916	0.936	0.958
	Truck	6	5	5	1	0	1	0.833	0.907	0.833
	Motorbike	1	1	1	0	0	1	1	1	1
	Bus	3	3	3	0	0	1	1	1	1
4	Car	49	46	46	3	0	1	0.938	0.968	0.938
	Truck	10	8	8	2	0	1	0.8	0.888	0.8
	Motorbike	3	1	1	2	0	1	0.333	0.50	0.333
	Bus	2	1	1	1	0	1	0.5	0.666	0.5
5	Car	37	33	30	7	3	0.909	0.810	0.856	0.891
	Truck	9	7	7	2	0	1	0.777	0.875	0.777
	Motorbike	0	0	0	0	0	0	0	0	0
	Bus	1	2	1	0	1	0.5	1	0.666	0.5
6	Car	50	47	45	5	2	0.957	0.9	0.927	0.94
	Truck	13	12	12	1	0	1	0.923	0.959	0.923
	Motorbike	3	2	2	1	0	1	0.666	0.799	0.666
	Bus	3	3	2	0	1	0.666	1	0.799	1

Figure 13. Tables that represent the counting results for the six lanes of the intersection.

Figure 15 shows the representation of data sent by our system in the form of a pie chart. The data represent the percentage of each path taken by the vehicles on 8 December 2022.

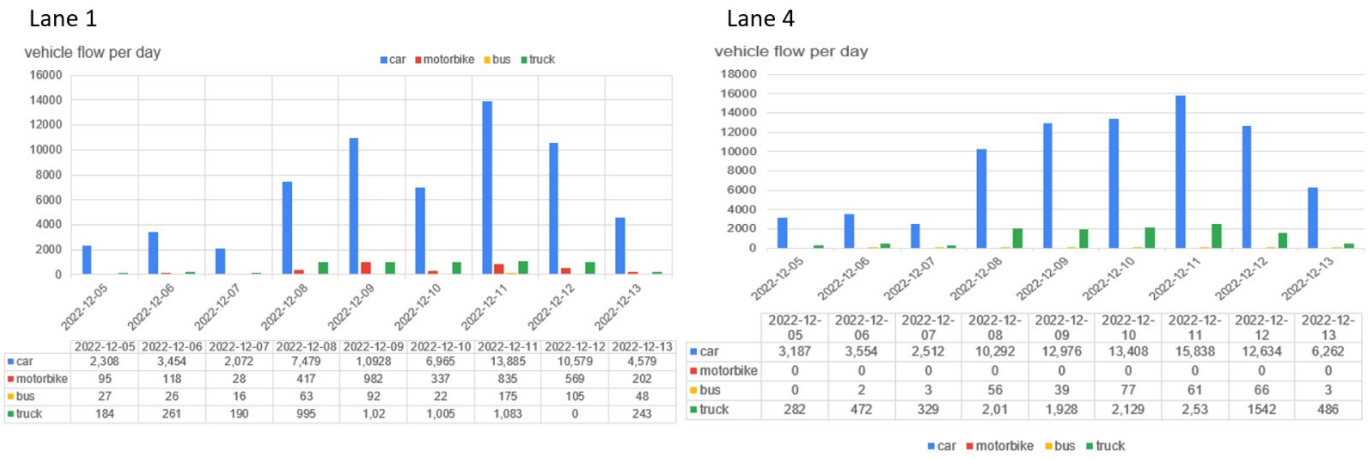
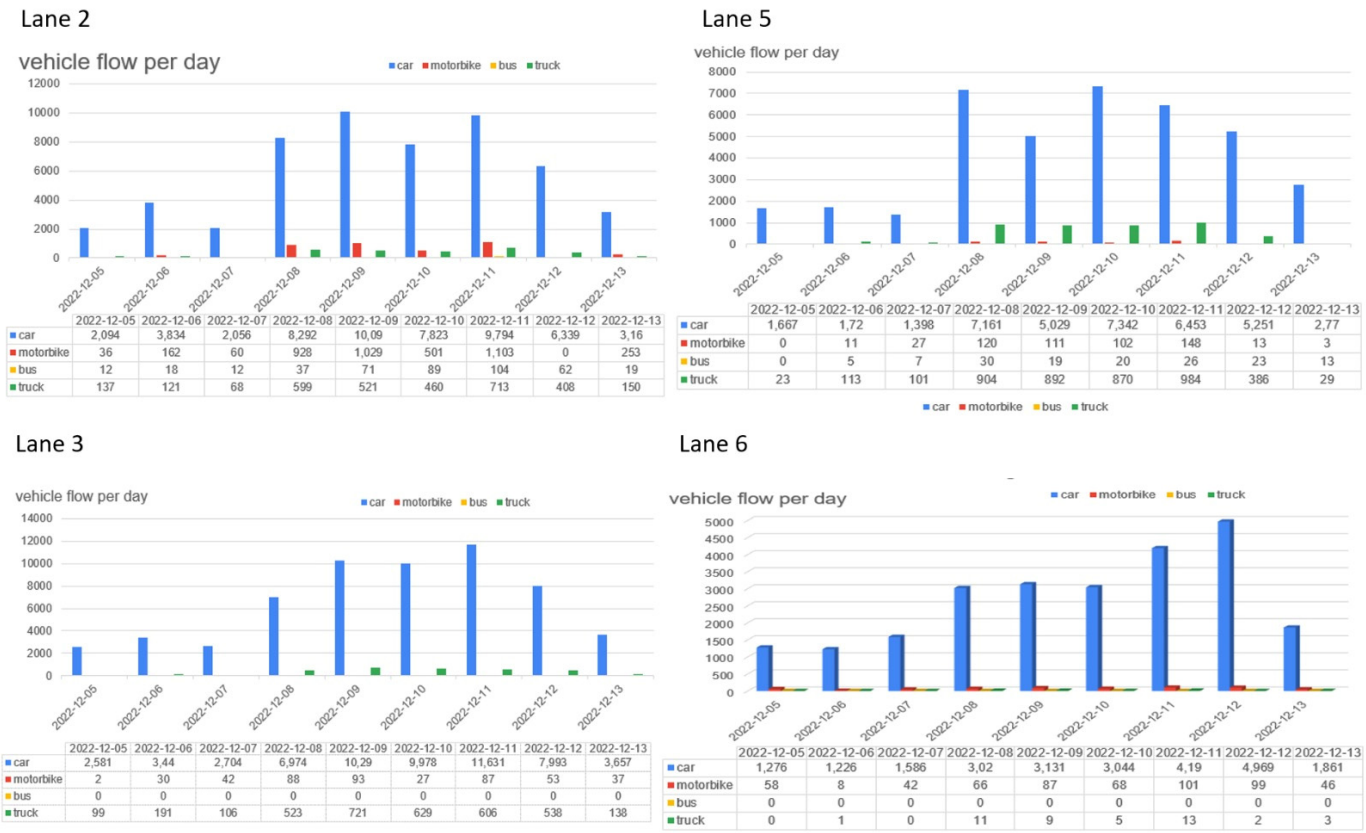


Figure 14. *Cont.*



Lane 2	2022-12-05	2022-12-06	2022-12-07	2022-12-08	2022-12-09	2022-12-10	2022-12-11	2022-12-12	2022-12-13
car	2,094	3,834	2,056	8,292	10,09	7,823	9,794	6,339	3,16
motorbike	36	162	60	928	1,029	501	1,103	0	253
bus	12	18	12	37	71	89	104	62	19
truck	137	121	68	599	521	460	713	408	150

Lane 5	2022-12-05	2022-12-06	2022-12-07	2022-12-08	2022-12-09	2022-12-10	2022-12-11	2022-12-12	2022-12-13
car	1,667	1,72	1,398	7,161	5,029	7,342	6,453	5,251	2,77
motorbike	0	11	27	120	111	102	148	13	3
bus	0	5	7	30	19	20	26	23	13
truck	23	113	101	904	892	870	984	386	29

Lane 3	2022-12-05	2022-12-06	2022-12-07	2022-12-08	2022-12-09	2022-12-10	2022-12-11	2022-12-12	2022-12-13
car	2,581	3,44	2,704	6,974	10,29	9,978	11,631	7,993	3,657
motorbike	2	30	42	88	93	27	87	53	37
bus	0	0	0	0	0	0	0	0	0
truck	99	191	106	523	721	629	606	538	138

Lane 6	2022-12-05	2022-12-06	2022-12-07	2022-12-08	2022-12-09	2022-12-10	2022-12-11	2022-12-12	2022-12-13
car	1,276	1,226	1,586	3,02	3,131	3,044	4,19	4,969	1,861
motorbike	58	8	42	66	87	68	101	99	46
bus	0	0	0	0	0	0	0	0	0
truck	0	1	0	11	9	5	13	2	3

Figure 14. Results of the count for the period of 5 December 2022 to 13 December 2022.

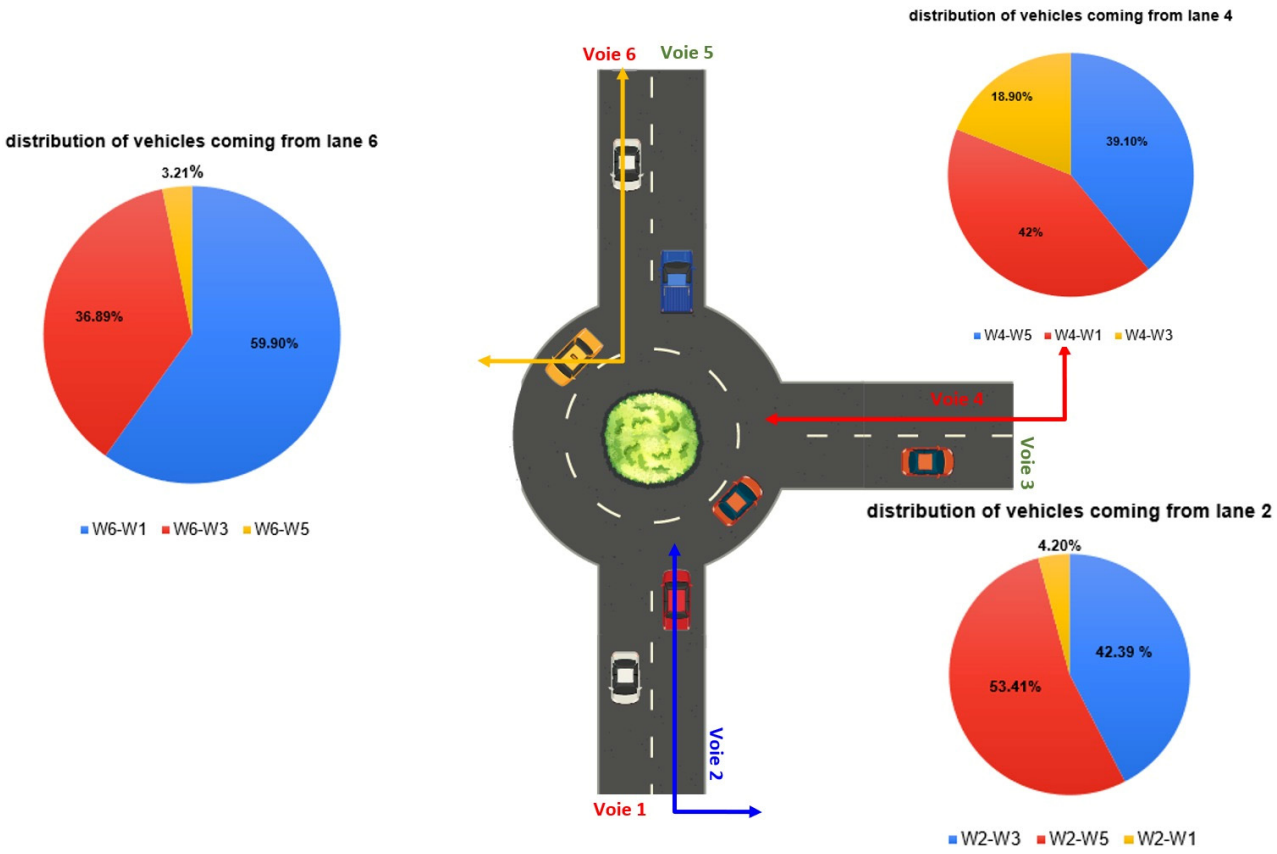


Figure 15. Pie charts represent the percentage of each path taken by the vehicles on 08/12/2022.

5. Discussion

Traffic management plays a crucial role in tracking the growth of cities such as Bouskoura, and our system has demonstrated its reliability in the experimental phase. The data collected are a valuable tool that helps the municipality make informed decisions regarding changes to the infrastructure, as well as to better understand the impact of opening of a new shopping center in the city. Specifically, our system has helped the municipality of Bouskoura to calculate, by class, the number of vehicles that pass through the roundabout. It can also help to determine the percentage of vehicles from each lane, which has been instrumental in deciding what lanes to extend and how to manage traffic flow to support the economic growth of the city. Overall, our data-collection system is an invaluable tool for the Bouskoura municipality in making informed decisions about traffic management and infrastructure adaptation to better meet the needs of its growing population.

6. Conclusions

In summary, our proposed vehicle-counting and classification system, which is based on a combination of a customized YOLOv7-tiny model and the Deep SORT algorithm, offers several key advantages over existing methods. We demonstrated that our system accurately counts and tracks vehicles that are using different lanes of a roundabout intersection while also providing the percentage of the trajectories that are performed by the vehicles. Moreover, our system can be deployed on the Nvidia Jetson Nano card, which has high performance and low power consumption, making it suitable for real-world scenarios. However, our system has certain limitations in accurately detecting vehicles in complex scenarios that have high occlusion and are under different lighting conditions. Furthermore, the quality of the input video stream and the camera's distance from the intersection can affect our system's accuracy. To address these limitations and to further improve our system, we can explore additional deep-learning techniques. We can also consider using multiple cameras to capture a more comprehensive view of the intersection and to improve our system's trajectory-estimation accuracy.

Author Contributions: Conceptualization, I.E. and M.A.S.; methodology, I.E., M.A.S. and A.A.; software, I.E. and Y.A.; validation, I.E., M.A.S. and Y.A.; writing—review and editing, I.E., M.A.S. and A.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Acknowledgments: We wish to acknowledge the support of the LISAC Laboratory at the University Sidi Mohamed Ben Abdellah who worked in collaboration with Nextronic by Aba technology Company, who was the host of this work.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ghazali, W.N.W.B.; Zulkifli, C.N.B.; Ponrahono, Z. The Effect of Traffic Congestion on Quality of Community Life. *ICRP* **2019**, *2*, 759–766. [CrossRef]
2. Jiber, M.; Mbarek, A.; Yahyaouy, A.; Sabri, M.A.; Boumhidi, J. Road Traffic Prediction Model Using Extreme Learning Machine: The Case Study of Tangier, Morocco. *Information* **2020**, *11*, 542. [CrossRef]
3. Patro, K.K.; Allam, J.P.; Hammad, M.; Tadeusiewicz, R.; Pławiak, P. SCovNet: A skip connection-based feature union deep learning technique with statistical approach analysis for the detection of COVID-19. *Biocybern. Biomed. Eng.* **2023**, *43*, 352–368. [CrossRef] [PubMed]
4. Pedada, K.R.; Rao, B.; Patro, K.K.; Allam, J.P.; Jamjoom, M.M.; Samee, N.A. A novel approach for brain tumour detection using deep learning based technique. *Biomed. Signal Process. Control.* **2023**, *82*, 104549. [CrossRef]
5. Shashirangana, J.; Padmasiri, H.; Meedeniya, D.; Perera, C.; Nayak, S.R.; Nayak, J.; Vimal, S.; Kadry, S. License plate recognition using neural architecture search for edge devices. *Int. J. Intell. Syst.* **2021**, *37*, 10211–10248. [CrossRef]
6. Padmasiri, H.; Shashirangana, J.; Meedeniya, D.; Rana, O.; Perera, C. Automated License Plate Recognition for Resource-Constrained Environments. *Sensors* **2022**, *22*, 1434. [CrossRef] [PubMed]

7. Hunkeler, U.; Truong, H.L.; Stanford-Clark, A. MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks. In Proceedings of the 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08), Bangalore, India, 6–10 January 2008; pp. 791–798. [CrossRef]
8. Uday, P.; Shikha, B.S.; Uday, P.; Shubham, S. Using YOLO V7: Development of Complete VIDS Solution Based on Latest Requirements to Provide Highway Traffic and Incident Real-Time Info to the ATMS Control Room Using Artificial Intelligence. *SSRN* **2022**, 4313791. Available online: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4313791 (accessed on 20 December 2022).
9. Mansour Mohamed, A.S.M.M.; Rashid, M.M. Video-Based Vehicle Counting and Analysis using YOLOv5 and DeepSORT with Deployment on Jetson Nano. *Asian J. Electr. Electron. Eng.* **2022**, *2*, 11–20. Available online: https://journals.alambiblio.com/ojs/index.php/ajoeee/article/view/34 (accessed on 1 March 2023).
10. Dinh, D.-L.; Nguyen, H.-N.; Thai, H.-T.; Le, K.-H. Towards AI-Based Traffic Counting System with Edge Computing. *J. Adv. Transp.* **2021**, *2021*, 5551976. [CrossRef]
11. Valladares, S.; Toscano, M.; Tufiño, R.; Morillo, P.; Vallejo-Huanga, D. Performance Evaluation of the Nvidia Jetson Nano Through a Real-Time Machine Learning Application. In *Intelligent Human Systems Integration*; Russo, D., Ahram, T., Karwowski, W., Di Bucchianico, G., Taiar, R., Eds.; Springer: Cham, Switzerland, 2021; Volume 1322. [CrossRef]
12. Han, Y.; Oruklu, E. Traffic sign recognition based on the NVIDIA Jetson TX1 embedded system using convolutional neural networks. In Proceedings of the 60th International Midwest Symposium on Circuits and Systems (MWSCAS), Boston, MA, USA, 6–9 August 2017; pp. 184–187. [CrossRef]
13. Wang, Z.; Bai, B.; Xie, Y.; Xing, T.; Zhong, B.; Zhou, Q.; Meng, Y.; Xu, B.; Song, Z.; Xu, P.; et al. Robust and Fast Vehicle Turn-counts at Intersections via an Integrated Solution from Detection, Tracking and Trajectory Modeling. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Seattle, WA, USA, 14–19 June 2020; pp. 2598–2606. [CrossRef]
14. Jiang, K.; Xie, T.; Yan, R.; Wen, X.; Li, D.; Jiang, H.; Jiang, N.; Feng, L.; Duan, X.; Wang, J. An Attention Mechanism-Improved YOLOv7 Object Detection Algorithm for Hemp Duck Count Estimation. *Agriculture* **2022**, *12*, 1659. [CrossRef]
15. Wang, Y.; Wang, H.; Xin, Z. Efficient Detection Model of Steel Strip Surface Defects Based on YOLO-V7. *IEEE Access* **2022**, *10*, 133936–133944. [CrossRef]
16. DEEP LEARNING. YOLOv7: The Most Powerful Object Detection Algorithm (2023 Guide). Available online: https://viso.ai/deep-learning/yolov7-guide/#:~:text=Compared%20to%20PP%2DYOLOE%2DL,or%20106%25%20faster%20inference%20speed (accessed on 25 March 2023).
17. WongKinYiu. YOLOv7 Official GitHub repository. Available online: https://github.com/WongKinYiu/yolov7 (accessed on 1 December 2022).
18. Lamouik, I.; Yahyaouy, A.; Sabri, M.A. Model Predictive Control for Full Autonomous Vehicle Overtaking. *Transp. Res. Rec. J. Transp. Res. Board* **2023**, *2677*, 1193–1207. [CrossRef]
19. Mandal, V.; Adu-Gyamfi, Y. Object Detection and Tracking Algorithms for Vehicle Counting: A Comparative Analysis. *J. Big Data Anal. Transp.* **2020**, *2*, 251–261. [CrossRef]
20. Xiang, X.; Zhai, M.; Lv, N.; El Saddik, A. Vehicle Counting Based on Vehicle Detection and Tracking from Aerial Videos. *Sensors* **2018**, *18*, 2560. [CrossRef] [PubMed]
21. Nvidia Corporation. Jetson NANO Module. Available online: https://developer.nvidia.com/embedded/jetson-nano (accessed on 2 January 2023).