TRAFFIC DENSITY ESTIMATION USING COMPUTER VISION

E FREESE and AJ HOFFMAN

School of Electrical, Electronic and Computer Engineering, North-West University,
Private Bag X6001, 2520, Potchefstroom, South Africa
Email: ettienefreese9@gmail.com

ABSTRACT

This paper describes a traffic density estimation system (TDES) that contributes to the field of intelligent transportation systems by providing the necessary queue length information to enable the optimal control of traffic lights. Modern computer vision techniques utilize Convolutional Neural Networks (CNN) to determine traffic queue lengths at intersections, including lane detection to ensure that only relevant vehicles are counted. We applied a trade-off study to different CNN models based on speed, accuracy, and implementation efficiency and found that the YOLOv8 model slightly outperformed the R-CNN and SSD architectures. Intelligent traffic control methods then apply Reinforcement Learning (RL) to this data to determine optimal cycle lengths for the different phases of traffic lights. We used the Simulation of Urban Mobility (SUMO) software to simulate a simplified traffic system consisting of traffic lights that manage different traffic levels as part of the daily and weekly traffic cycle. The simulated system allocated priority to emergency vehicles and buses over normal passenger vehicles. Our simulations indicated that the combination of the TDES and RL traffic control outperforms traditional fixed-time traffic control methods for all traffic densities. The results showed that the TDES achieved a 70% mean average precision and a maximum 50% waiting time error. For a variety of traffic scenarios applied to a single intersection architecture, the improved traffic control method reduced average vehicle waiting time by almost 90%.

1. INTRODUCTION

As cities expand and the population grows, an increase in traffic density leads to traffic congestion, unwanted delays and pollution (Prasad, Kapadni et al., 2019). This results in a decline in productivity and has significant negative impacts on society and the economy (Chung, Le et al., 2022). Traditional methods, including fixed-time delays of traffic lights or manual control by traffic officials fail to address this problem. There is thus a necessity for a smart, adaptive, and optimized control system to enhance the efficiency of traffic flow.

Such a smart traffic system must estimate the traffic density and regulate traffic lights accordingly (Prasad, Kapadni et al., 2019). In this paper, traffic density is defined as the number of vehicles present in a defined area, with their accumulated waiting times, at a given instant. Computer vision techniques in combination with closed-circuit television (CCTV) datasets provides a low-cost surveillance approach for vehicle classification, detection, and tracking (Chung, Le et al., 2022). Challenges include the presence of vehicle shadows, occlusions of vehicles, and environmental conditions such as rain, and bad lighting, which decay performance. Most existing methods are restricted to specific road scenarios and require on-site calibration procedures (Asha, Narasimhadhan et al., 2018).

Traditional methods to measure traffic flow rates used underground inductive loops, pneumatic road tubes, and temporary manual counts. These approaches cannot be implemented in large areas due to its high cost, road degradation, and implementation difficulties such as traffic disruption (Shirazi, Morris et al., 2015). Another alternative use electronic sensors placed beside the road to collect in- formation about traffic density (Malose John, Chunling et al., 2023). Further alternatives include spatial-based sensors such as GPS, Bluetooth, and infrared technology (Raj, Bahuleyan et al.). Finally, traffic prediction machine learning techniques can also be considered, using linear regression models and Kalman filtering techniques (Raj, Bahuleyan et al.).

Compared to these traditional methods, vision-based sensors use images obtained from CCTV to estimate traffic density, detect vehicles across different lanes, classify and track them according to their class, estimate speed, queue length, and movement direction (Muhammad, Muhammad Umar et al., 2021). Computer vision sensors have significant advantages over traditional methods, such as low installation and maintenance costs, which solely need cameras that are preinstalled on junctions and demand no other type of sensor (Fedorov, Nikolskaia et al., 2019; Prasad, Kapadni et al., 2019). It is simple to use and maintain, and can be upgraded simply with a software update, without needing to replace the entire sensor.

The three main tasks involved when estimating traffic density include object detection (OD), classification, and tracking (Fedorov, Nikolskaia et al., 2019). The computer vision techniques for traffic estimation includes background subtraction, edge detection, Histogram of Oriented Gradients (HOG), Local Binary Pattern (LBP), and deep learning neural networks (DNN), each with its advantages and disadvantages (Prasad, Kapadni et al., 2019).

(Shengli, Jianmin et al., 2006) proposed a background subtraction method, which subtracts a reference image when no traffic is present on the road from subsequent frames having traffic. The result leads to a frame with solely traffic vehicles; the traffic density is associated with the number of pixels in the resulting frame (Prasad, Kapadni et al., 2019). The drawback of background subtraction strategies results from changes in environmental conditions. If the reference image is taken in the daylight, the accuracy of the prediction decreases as the daylight changes (Prasad, Kapadni et al., 2019). Similarly, this method does not perform well when the traffic conditions get heavy. The predictions merge different vehicles during partial occlusions, leading to the wrong bounding box predictions and lower vehicle counts (Asha, Narasimhadhan et al., 2018).

Another method is based on deep learning models, which were developed as classical machine learning (ML) algorithms have difficulties processing large amounts of data (Dilek and Dener, 2023). Deep neural networks (DNN) consist of multiple layers including an input layer, several hidden layers, and an output layer. The convolutional neural network (CNN) displayed in Figure 1 is the deep learning architecture that is most widely used for feature extraction in computer vision (Malose John, Chunling et al., 2023).

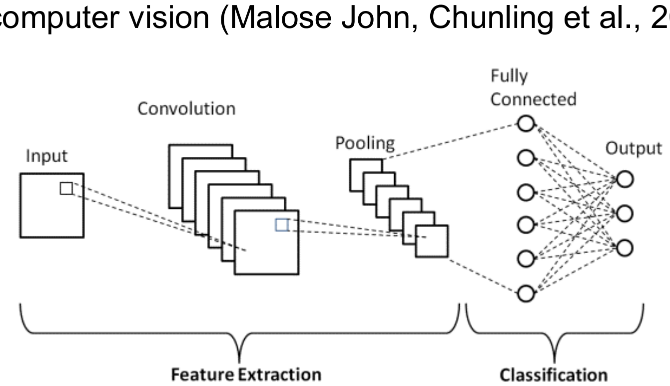


Figure 1: CNN

1.1 Aim of Paper

This paper estimates traffic density at traffic light intersections using computer vision to determine queue lengths in each lane approaching the intersection. Traffic density is defined as the number of vehicles present within a given length of road at a specific point in time. The queue lengths are then used to determine optimal cycle lengths for the traffic lights to minimize traffic delay times. Longer queue lengths will be used to lengthen the green cycle times of the corresponding lanes compared to other lanes in order to establish more balanced traffic flows and in the process improve overall traffic system performance.

1.2 Problem Statement

The literature showed that traditional methods to control traffic are very inefficient as they either do not adapt to changing conditions or are manpower intensive and subject to human error. These methods can be significantly improved by using intelligent methods such as computer vision to estimate the traffic density. This measurement can then be used as input for a more intelligent traffic control algorithm

1.3 Scope of Paper

The paper describes the development of an object detection and tracking model that estimates the traffic density that is used as input to a smart traffic light controller that improves traffic flows.

2. MODELING METHODS USED

2.1 Neural Networks

Neural networks are trained by adjusting the weights between units, also called neurons. In each neuron, a scalar input is transmitted through a connection that multiplies it by the scalar weight and then adds a bias value, which then applies a transfer function to form an output (Raj, Bahuleyan et al.). During training these weights are adjusted through an iterative process, known as backpropagation, so that specific inputs results in a specific output that minimises the loss function, calculated as the difference between the output and the target.

To obtain an accurate model, it is very important that the training data contains all possible variations that are of importance to model outcomes. This includes variety in daylight, vehicle density, type of vehicle, and angle relative to the camera. Although this method does provide very accurate results, any deep learning (DL) based method is a resource-intensive and computationally costly task. Rather than training the model from scratch, an alternative will be to explore the use of transfer learning approaches, which are based on the idea of using existing trained models (Raj, Bahuleyan et al.). These pre-trained models have initial weights and biases that change as new training progresses, which minimises the training time and computational power required.

Before any model can be trained or changes to an existing model are made, its hyperparameters should be adjusted based on some performance metrics. Adjusting a model's hyperparameters is critical to ensure the model performs better over each training period also known as epochs. During each epoch, the model uses batches to improve its training accuracy by observing multiple data simultaneously. Another hyperparameter that should be specified is the learning rate of the model. This specifies the step size of each iteration during training while moving towards a minimum in the loss function.

The total dataset is split between training (80%), validation (10%), and testing (10%) data. The validation data is used during training to observe the success rate of the training process while preventing overfitting the training data. Finally, after a model has been trained, some performance metrics during the evaluation phase are used to decide whether additional training is required. Evaluation is typically done on the unseen testing data of the dataset. For example, intersection over union (IOU) is an evaluation metric calculated as the area of overlap divided by the union of a predicted object and the ground truth.

2.2 Pattern Recognition Algorithms

In the field of object detection two main algorithms are utilised: single-stage and two-stage detectors. Single-stage detectors such as Single Shot Detector (SSD) and You Only Look Once (YOLO) predict the bounding boxes around the object together with the class it is associated with within a single network pass (Farhadi, Girshick et al., 2016). In contrast, two-stage detectors first identify the regions of interest (ROI) and then uses a classifier to identify the objects (Farhadi, Girshick et al., 2016). Two-stage detectors such as R-CNN family networks tend to be slower compared to its alternative, although they tend to be more accurate.

A series of architectures have been considered and improved over time. Regions with CNN features (R-CNN) used a selective search algorithm and feed candidate ROI to a CNN for classification, as shown in Figure 2 (Olorunshola, Jemitola et al., 2023). It was later improved using a new Fast R-CNN algorithm, which outperformed it in terms of speed and accuracy, however, still relied on an external region proposed algorithm. Finally, it was improved by making the region proposed network (RPN) fully convolutional, as seen in Figure 3. However, the R-CNN family of networks still proved to be very slow reaching real-time speeds of less than 10 fps (Olorunshola, Jemitola et al., 2023).

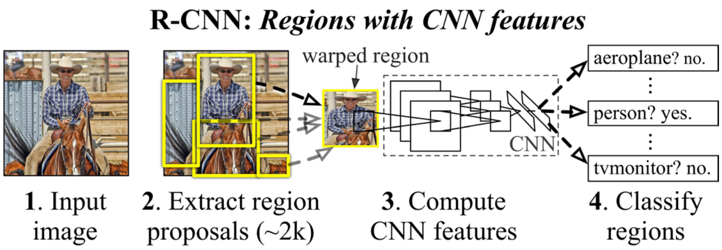


Figure 2: Original R-CNN architecture

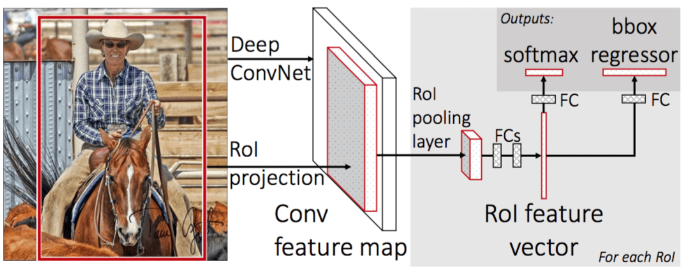


Figure 3: Fast R-CNN architecture

Both SSDs and YOLO use a one-stage detector algorithm that solves the slow computation problem. These detectors treat object detection as a regression problem, learning bounding box coordinates of the images and labelling corresponding class probabilities (Huang, Rathod et al., 2017). Figure 4 shows the YOLO algorithm first introduced by (Farhadi, Girshick et al., 2016) in 2015, which achieved 45 FPS on GPU, similarly, Figure 5 shows the SSD algorithm which is close competition.

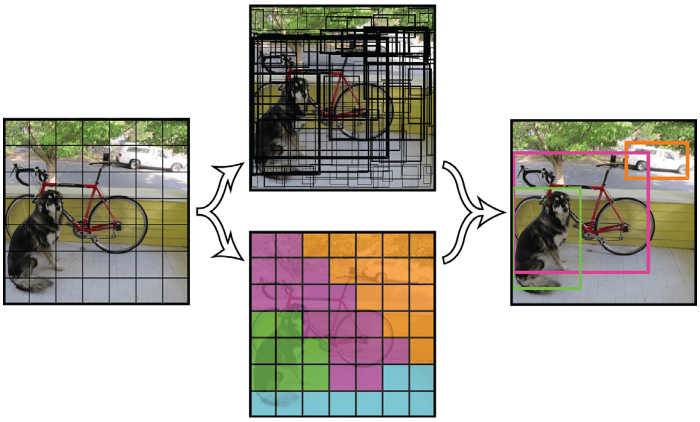


Figure 4: YOLO architecture

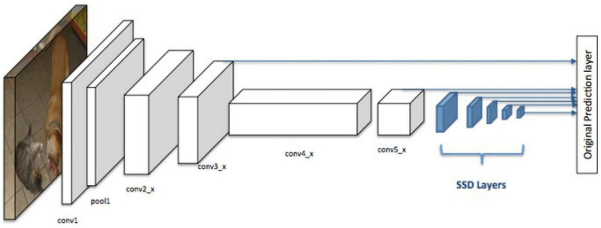


Figure 5: SSD architecture

2.3 Tracking Algorithms

Tracking algorithms use the location of the object's bounding box, which can be user-specific or the output of an object detector. The tracking algorithm assigns a tracking ID to each detection and follows the object over each frame (Asha, Narasimhadhan et al., 2018). There are two categories of tracking algorithms, namely Single object tracking (SOT) and Multiple object tracking (MOT). In SOT, a single object is tracked from the beginning, while in MOT, several objects are detected and tracked from one frame to the other Two popular examples of SOT algorithms include Kalman Filtering and Particle Filtering, whereas SORT, DeepSORT, and ByteTrack are state-of-the-art MOT algorithms, as shown in Figure 6.

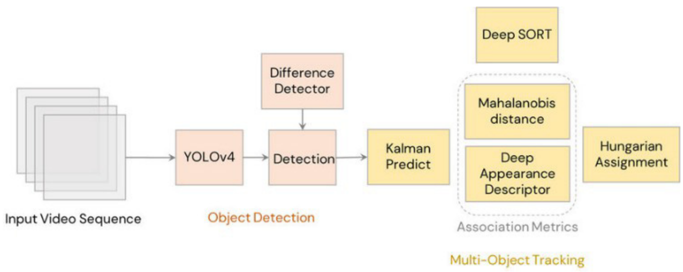


Figure 6: MOT tracking algorithm

These algorithms extend Object Detection by not only identifying objects in a series of images but also maintaining unique instance IDs for each object throughout a video sequence. MOT includes two main stages: the first stage consists of Object Detection, where objects are identified. The second stage includes Instance association, where information over time is combined to generate trajectories for individual objects, ensuring consistent tracking IDs across frames.

One of the major challenges of traffic density estimation is dealing with occlusions, where vehicles overlap or pass each other, making it difficult to maintain accurate trajectories. The MOT algorithms maintain a consistent understanding of each vehicle's movement, even when some vehicles are partially or fully obscured.

Scalability and adaptability to varying traffic conditions are critical for these tracking algorithms. Scaling is important to handle fluctuations in traffic density, while additionally, the algorithms need to adapt to changes in vehicle speeds, as vehicles may slow down or speed up depending on traffic conditions. It can thus be concluded that different tracking algorithms need to be compared and chosen based on performance metrics such as speed, accuracy, and implementation complexity.

2.4 Reinforcement Learning

For optimising traffic light cycles after the queue lengths have been established, we used another type of machine learning algorithm that is called reinforcement learning (RL). RL is different from supervised learning, such as previously discussed, which uses a training set of labelled examples, provided by some experts. It is also different from unsupervised learning, as it tries to maximise a reward function, instead of finding some hidden structure in collections of unlabelled data (Sutton and Barto, 2020).

RL focuses to optimise the decisions made by agents and maximise their performance, controlled by constant reward feedback. During the process, the agent should decide on the best possible action, based on its current state. This mathematical modelling process uses discrete time steps, where the agent takes new actions that result in new environmental states; this is called the Markov Decision process (Sigaud & Buffet, 2010).

The agent is not told which actions must be performed but instead must discover which actions build up the best reward. For traffic control these actions include changing the traffic signals based on the traffic densities at various lanes of the intersections. Through continuous trial and error attempts, the agent builds a set of rules and policies, to help it decide which action to perform next while maximising its cumulative reward. These agents may also explore new environmental states, which can result in better or worse rewards; this is known as the exploration-exploitation trade-off.

In RL there are model-free and model-based algorithms. Model-based tries to understand the environment's rules, while model-free only remembers what decisions give them good rewards. The last factor to discuss in RL models is single and multiple-agent algorithms. An RL model may incorporate multiple agents, that also consider the states and actions of its neighbours, which can make the entire system more complex, but also more intelligent. For simplicity, we only consider single-agent RL in this paper.

3. REQUIREMENTS

3.1 Functional Requirements

The following functional requirements were identified to satisfy the user needs to optimise a traffic system:

- The system shall make use of a pre-trained object detection (OD) model to detect vehicles at an intersection.
- The system shall define each lane of an intersection so that this solution can be replicated on any intersection with minimal manual labour.

- The OD model shall use a custom or manually gathered and annotated dataset of different vehicles in different conditions for training.
- The system must receive good-quality video footage, from cameras that are installed in suitable positions at intersections.
- The OD model must identify different types of vehicles, as each type will influence the quality of traffic density estimation.
- The system shall recognise different vehicles in different traffic density scenarios and times of day.
- The system shall detect if a vehicle is moving or stationary, as this will allow more accurate traffic density estimation, and ignore irrelevant data, such as far away moving vehicles.
- The system must use the output of the OD model to track each vehicle that it detected and count these objects if they are located in a defined area.
- The system shall count vehicles that pass the intersection, in both directions, for green, yellow, and red lights.
- The system shall detect occlusions, such as smaller vehicles partly obscured by larger ones such as trucks or buses, as part of the contribution to the overall traffic density.
- Output of the traffic density estimation shall be displayed on a suitable graphical user interface (GUI), for the operator and observer to inspect the performance of this system on any video footage.

3.2 Performance Requirements

From the functional requirements, the following performance requirements were derived:

- The performance of object detection is described in term of mean average precision (mAP), which is the number of correctly classified objects (in this case vehicles) divided by the total number of detected objects (some of which may not be actual vehicles).
- A related performance measure is recall, which is the number of correctly classified objects divided by all objects present (ground truth).
- As mAP does not take ground truth into account, it is necessary to define the Intersection over union (IOU), which is an evaluation metric calculated as the area of overlap between predicted and ground truth (i.e. the number of correctly detected vehicles) divided by the union of predicted and the ground truth (the total number of predicted plus actual vehicles present). mAP can then be defined for a specific value of IOU, e.g. mAP50 is the mean average precision when IOU = 50%.
- mAP50-95 is the average of the mean average precision calculated at varying IOU thresholds, ranging from 0.50 to 0.95. It gives a comprehensive view of the model's performance across different levels of detection difficulty.
- As we could not find any guideline from literature for the required mAP within this context, we determined through trial and error that the OD model must perform with at least 70% mAP for the entire video footage provided and perform at 30 frames per second, so that it can be used in real-time using CCTV surveillance.
- The tracking software, that tracks detected objects from one frame to the next, shall track at least 80% of all objects detected and assign a unique ID to each until the object moves out of frame.

- The primary performance measure for the traffic system to be optimised is the average accumulated waiting time of vehicles that are stationary at traffic lights. As we also could not find any guideline from literature for the required waiting time error, we set a requirement of 25% relative error (waiting time error divided to total waiting time) between measured and actual waiting times.
- The quality of video footage from cameras needs to be at least 1280x780 pixels resolution, with the camera angle showing all lanes and at least a queue length of five vehicles, depending on the traffic popularity of the intersection. In an ideal case the cameras must collected aerial views of traffic to prevent occlusions, i.e. where some vehicles are obscured by other vehicles from the viewing angle of the camera. In this case we had to accept the limitations of the cameras that were installed for the purpose of pedestrian safety observations.
- The GUI shall be designed to allow the user to upload video footage and observe the results at 6+ fps for 100% of the footage, with no errors. The minimum required 6 fps was set due to the program needing to only process every fifth frame, and thus operate at 30 Hz, which is the minimum live fps of most video footage. Recognition of traffic lanes shall be estimated to ground truth within 10% of actual lane width.

4. DESIGN

4.1 Architectural Design

From the analysis done in the previous section, a formal architecture diagram was developed as shown in Figure 7. The traffic density estimation program, which interfaces with all functional units, consists of the main software program that lies centrally to the entire system.

4.2 Selection of Techniques

For computer vision object detection, the Convolutional Neural Network approach will be implemented as it provides high-accuracy detection and classification. This choice will allow additional features to be implemented in future software versions such as unique priority regulation on specific vehicle classes.

The network architecture is crucial as, in addition to mean average precision (mAP), speed and memory usage should also be considered. The R-CNN architecture outperforms SSD and YOLO in terms of accuracy but is slower and less efficient due to the complex architecture Both YOLO and SSD algorithms support easy implementation with libraries such as Ultralytics or OpenCV. These architectures provide very fast results while maintaining high precision. Smaller YOLO models outperform SSD in terms of speed while sacrificing some accuracy, while some SSD models outperform older YOLO models (Olorunshola, Jemitola et al., 2023). YOLO further provides easy integration with tracking algorithms. From the analysis, it can be concluded that YOLOv8 will be used to solve the object detection and classification problem.

The evaluation metrics for object tracking include Multiple Object Tracking Accuracy (MOTA), Multiple Object Tracking Precision (MOTP), and Processing Speed (Abouelyazid 2023). These evaluation metrics measure the tracking performance by accounting for false positives, false negatives, and number of ID switches, which must be as low as possible. It also measures the localization precision by taking the difference between the predicted bounding boxes and the ground truth. Mostly Tracked (MT) metric is defined as the percentage of ground truth trajectories tracked for over 80% of their lifespan, and needs to be maximized, while the Mostly Lost (ML) metric is defined as trajectories tracked for less than 20% of their lifespan, and needs to be minimized. Based on our analysis it was concluded that ByteTrack outperforms the SORT and DeepSORT algorithms on most performance metrics.

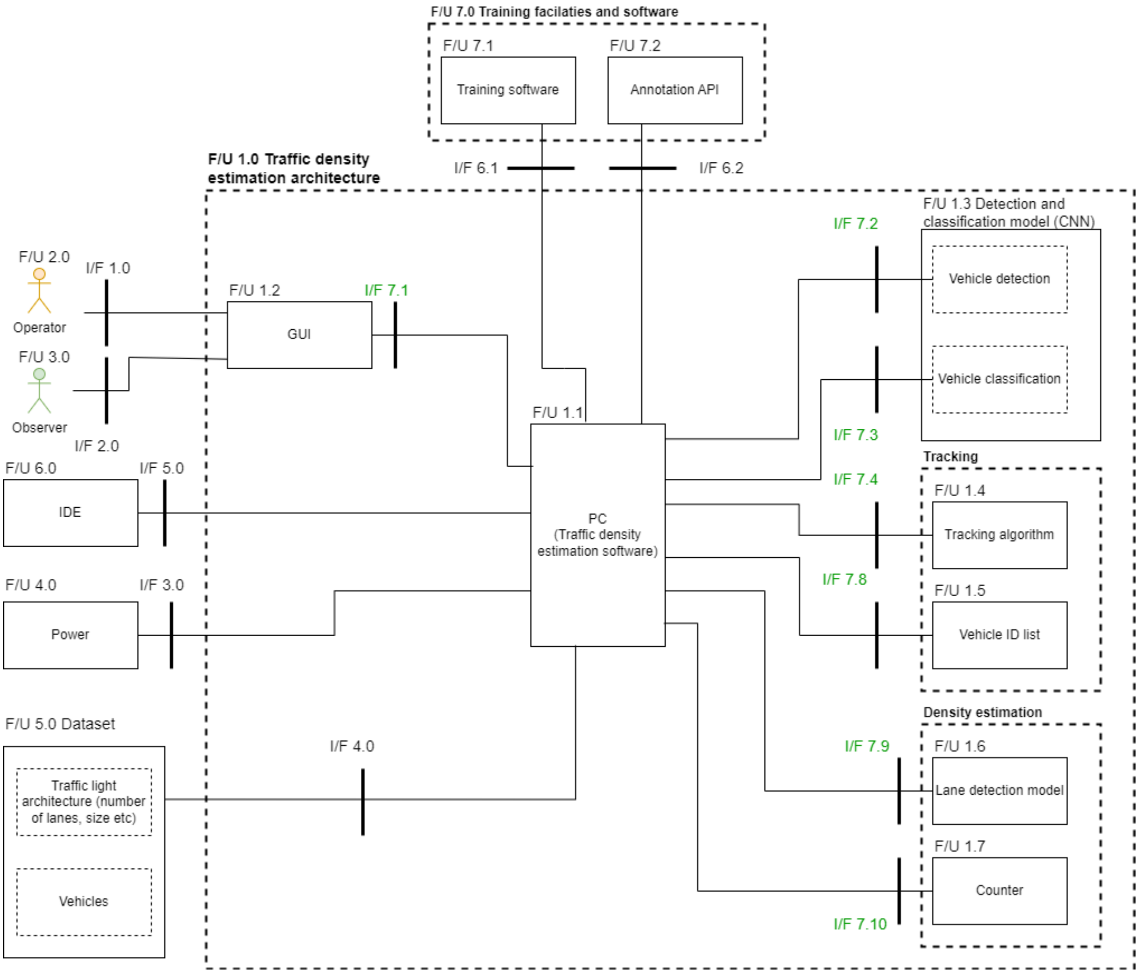


Figure 7: System architecture

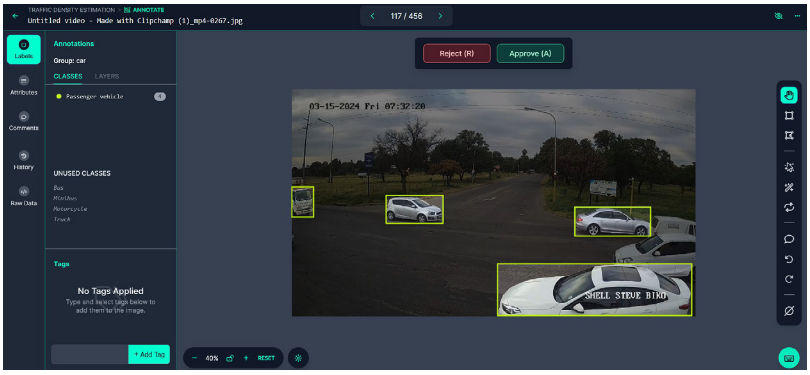
5. PERFORMANCE EVALUATION

The performance of each functional unit will determine the expected impact of traffic density estimation on traffic flows. Once the performance of object detection, classification and tracking was measured we will perform a simulation to determine the need for further improvements to the OD and classification model.

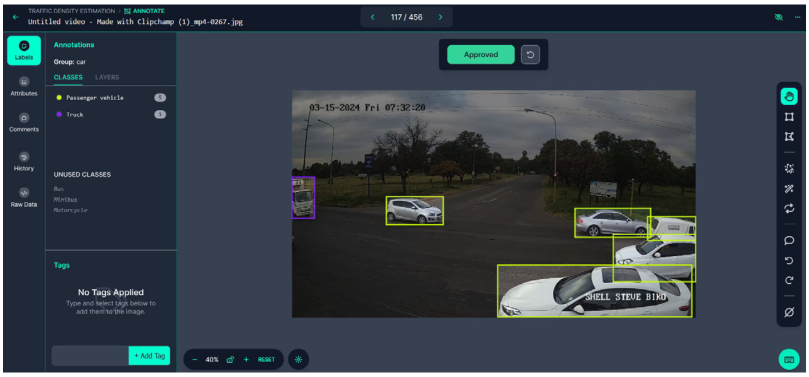
5.1 Test 1: Object Detection Model

The purpose of this test is to validate the performance of the method chosen to recognize all types of vehicles, with some being partly obscured. As per the performance requirements defined in 3.2 an average of at least 70% of all vehicles must be detected, with a 50% confidence level. An example of how the auto annotation model uses a trained model to predict the bounding box locations and classes is shown in Figure 8, which shows how additional predictions and classifications needed to be made before the image could be accepted to be part of the dataset.

Confusion matrices are used to indicate the accuracy of the classification of the models. The results for the best trained model are displayed in Figure 9. It achieved average classification accuracy of 75.8%, with the lowest classification accuracy of 60% for motorcycles. Most background objects such as pedestrians or cyclists (detections associated with no labelled class) were categorized as passenger vehicles, as this was the most populous vehicle type in the dataset.



(a) Auto annotation result before accepted



(b) Auto annotation result after accepted

Figure 8: Manual acceptance of auto annotation results

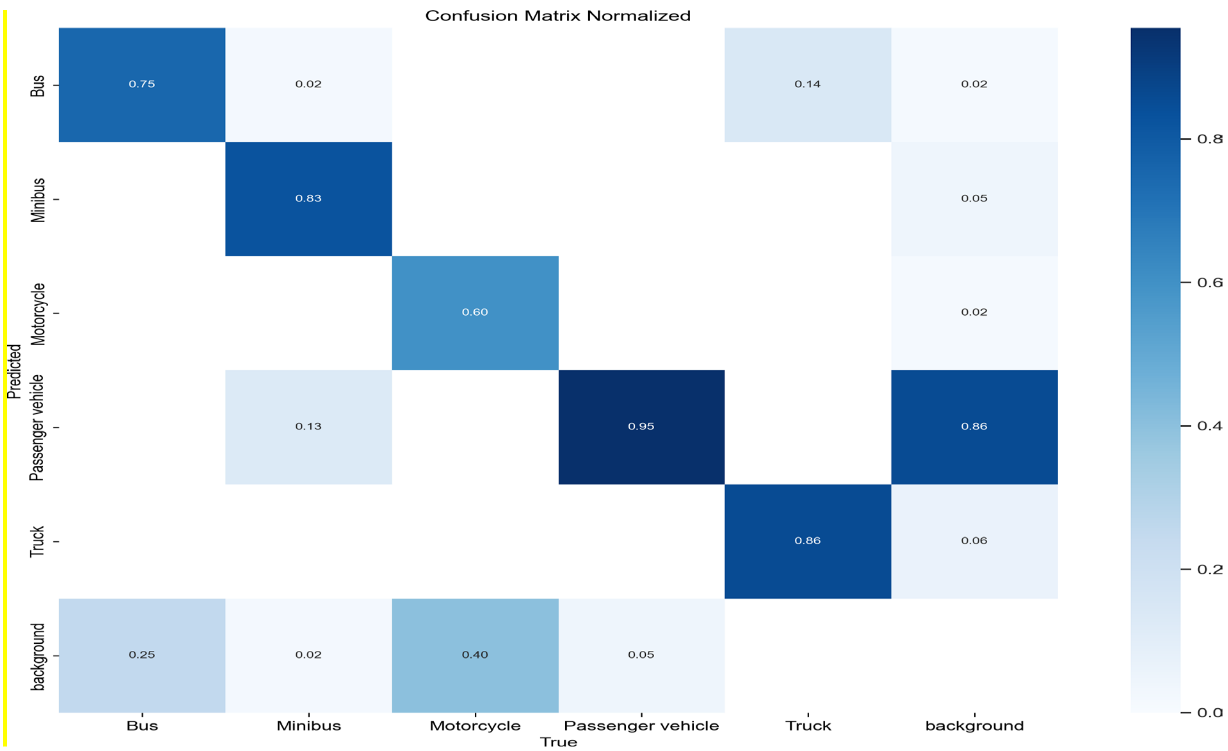


Figure 9: OD models confusion matrices

Figure 10 shows how model performance improved during training, with the x-axis showing the number of epochs during training and the y-axis the unit of measurement as fractions of 1. The top row of graphs display training results and the bottom row validation results. In the case of the graphs displaying training and validation loss it measures the size of the model error compared to a fixed reference error. In the case of the different metrics a value of 1 means perfect performance. The trained model reached close to 80% mAP50 and almost 70% mAP50-95 in the validation set. Furthermore, it can be noted that very little class loss is present.

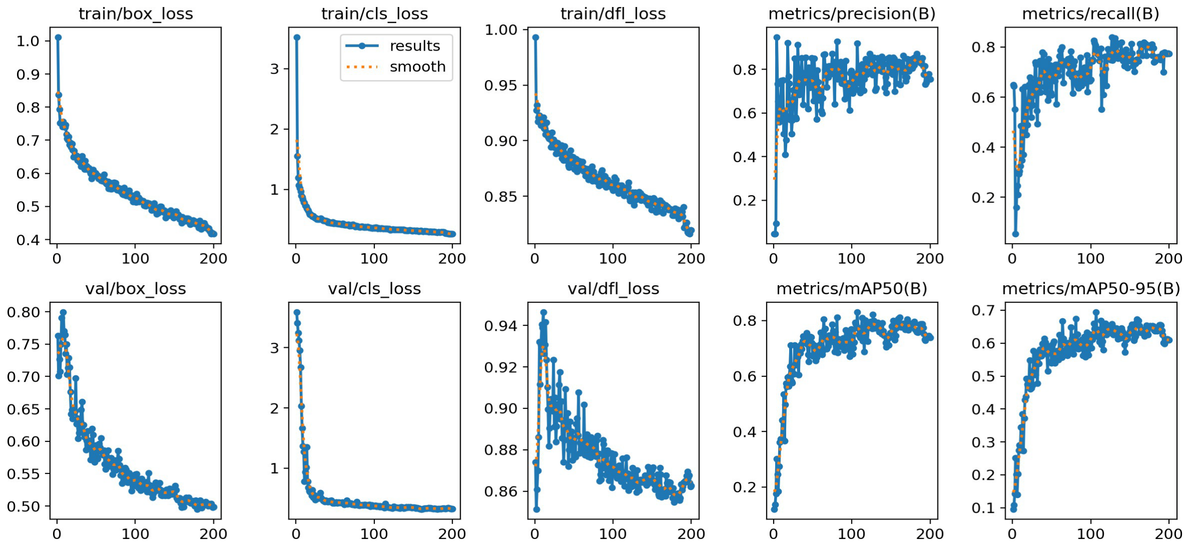


Figure 10: Comparison of OD models quantitative results

5.2 Test 2: Vehicle Tracking and Counting

Tracking each object is very important, as it impacts the accuracy of counting the number of vehicles within a lane. Each object is assigned an ID and keeps that ID until it is no longer present in the frame. 90% of vehicles that enter or are stationary in the designated area were identified and counted to determine the overall traffic density. Of vehicles being tracked less than 5% may lose their ID in the process.

Figure 11 confirms that the entire vehicle tracking algorithm performs as designed and that the functions in the system architecture were integrated successfully. All detected vehicles are tracked and counted separately within their uniquely defined lanes. The different classes (Bus, Minibus, Motorcycle, Passenger vehicle, Truck) were correctly identified for each lane. Finally, each unique tracking ID has a timer that calculates the vehicle's accumulated waiting time while the vehicle is stationary (S).

5.3 Waiting Time Measurement Accuracy

To confirm the waiting time function was implemented successfully a series of tests were performed. Three videos were used in these tests, Video 1 was recorded at 7 am with very good lighting conditions and moderate to high traffic density. Video 2 is recorded at the same intersection at 6 am with worse lighting conditions and less traffic density. Finally, video 3 was recorded at a different smaller intersection, with moderate traffic density conditions. A few random tracking IDs were used to compare the measured vs true waiting times as well as if any tracking ID was lost. True values were measured using a stopwatch, using manual inspection, and timing of these output videos. We also measured the ID loss, which is cases where the identity of a detected vehicle was lost during the process of measuring waiting time.

(a) Identification of lanes

(b) Lanes stored in database

(c) Traffic Density Estimation example

Figure 11: Illustration of the detection of vehicles per lane

From Table 1 it can be noted that the average absolute and relative errors are 2.4s and 19.65% respectively, which is below the objective of 25%. The 7:3 ratio ID loss is mostly due to fully obscured vehicles for a certain period in which the Byte Track algorithm assigns a new ID to the detection thinking it is a new vehicle. This phenomenon can be prevented by obtaining better camera angles where bigger vehicles such as Trucks do not fully obscure all vehicles in the frames.

We used a two- sample student t-test and verified that there exist no significant differences between the actual and measured waiting times. The differences between the measured and true waiting times are close to normally distributed, the measurements are independent of each other, and finally, the variances of true and measured waiting times are similar.

Table 1: Vehicle waiting time measurements

Video num	Vehicle ID	Measured WT [s]	True WT [s]	Abs Error [s]	Rel Error [s]	ID Loss
Video 1	#2	11.9	11.0	0.9	8.18%	False
Video 1	#3	13.2	11.7	1.5	12.82%	False
Video 1	#21	21.5	23.8	2.3	9.66%	False
Video 1	#43	16.5	12.15	4.35	32.9%	True
Video 2	#13	6.7	4.72	1.98	41.95%	False
Video 2	#46	6.3	6.5	0.2	3.08%	True
Video 2	#98	17.7	16.38	1.32	8.06%	False
Video 2	#103	19.5	26.17	6.67	25.49%	False
Video 3	#17	5.0	7.1	2.1	29.58%	True
Video 3	#47	13.4	10.74	2.66	24.77%	False
				Avg: 2.4 Max: 6.7	Avg: 19.65% Max: 41.95%	F:T= 7:3

5.4 Optimization of Traffic Flows

This section describes the traffic control tests that were performed using the SUMO software as well as a Reinforcement learning algorithm using the TraCI and SUMO-RL libraries (Janota, Kalus et al. 2024). These simulations compare the performance of the two traffic control methods (FT or Fixed Time and RL or reinforcement learning) for different traffic density scenarios, also verifying additional features such as priority regulation and multiple intersections.

The conventional FT method allowed 33s for the green and red straight lights, 6s for the green and red turning lights, and finally 2s for the straight and turning yellow lights. The RL program was instructed to keep the cycle times between a minimum of 5s and a maximum of 50s for red and green lights, with the yellow lights remaining unchanged at 2s. The simulations thus used a minimum of 2 seconds of inter-green-time between the turning and opposing straight lanes. However, the inter-green-time between conflicting straight lanes (e.g., NS to EW) was effectively 10 seconds, as it included the green and yellow times of the turning lanes in between. This extended gap ensures safer transitions between high-speed straight traffic flows. While the shortest time cycles may exceed the safety limits as stated in the SA Road Traffic Signs Manual, the intention was to study theoretical behaviour over a range of values that includes the most extreme cases that may be encountered in practice.

Each route file containing different traffic densities for 100 000 steps (seconds) is up-loaded together with the network file to test both methods' responses for the different traffic scenarios. For each 25 000 steps the traffic densities for different directions are changed, to replicate practical scenarios of sudden traffic density changes in each direction, although the overall traffic density for each scenario is kept constant throughout the entire simulation.

Scenario 1: Single intersection FT vs perfect RL for different traffic densities

Four different traffic scenarios will be simulated, containing the original traffic density (100%), 10% of it, 30%, and 50% of the original data. The 100% scenario results in roughly 120 vehicles per hour on the busiest lanes of the intersection. For the other scenarios proportionally smaller numbers of vehicles were launched into the system by the simulator. Table 2 provides the mean accumulated waiting time per vehicle over 100K steps for the five scenarios. It can be noted that the RL model performs better for all traffic scenarios.

Table 2: Waiting time data for RL and FT at different traffic densities

Traffic Density:	10%	30%	50%	100%
RL	0.346 s	0.757 s	1.225 s	3.096 s
FT	4.325 s	11.42 s	17.03 s	40.442 s

Scenario 2: Single intersection FT vs contaminated RL simulations

The RL contaminations include a variation in the number of vehicles used by the RL algorithm due to variation in OD accuracy (i.e. measured vs actual vehicle counts); this was implemented using a random number generator and an OD accuracy threshold. Another contamination includes adding uniformly distributed noise to the accumulated waiting time (i.e. a WT error) of each vehicle, by varying the absolute maximum error. This simulation will allow us to observe for which OD accuracy the FT method outperforms the contaminated RL method (if any), and how the RL program responds to these contaminations.

From Table 3 it is clear that the model WT error improves as the OD accuracy increases. Using the results from Table 3 it can be concluded that a maximum WT error of around 50% and OD accuracy of around 70% results in a mean accumulated WT per vehicle of 4.5 seconds for the given traffic density and intersection design. This is still much superior to the FT method which was simulated to result in around 40 seconds of mean accumulated WT per vehicle, as seen in Table 2. The actual OD accuracy of the classifier, combined with RL optimization, will therefore reduce average waiting time by almost 90%.

Table 3: Average waiting times of the RL model results to different contaminations

	OD accuracy				
WT max abs err	10%	30%	50%	70%	100%
0%	8,8393	5,0299	4,5574	4,357	3,0962
25%	10,6888	5,3597	4,2584	4,2085	3,6113
50%	10,0883	5,4449	4,9604	4,508	3,5482
75%	11,2695	5,6493	5,18	4,122	3,5776
100%	9,3102	6,9868	5,0152	4,5413	3,7095

Scenario 3: Single intersection priority regulation

This scenario allocates different priorities to different classes (busses given high-priority vs passenger vehicles with low-priority). Apart from the addition of buses to the route file, the total traffic density remains the same as the 100% scenario. Furthermore, the RL program identifies each vehicle's class and multiplies the buses' accumulated waiting time by a factor of 50 to represent the estimated number of passengers per bus, while other classes' WT are kept unchanged. This causes the algorithm to think that lanes containing buses have more vehicles or vehicles that have waited a long time, thus assigning more priority to those lanes.

Table 4 shows the mean accumulated waiting times over all time steps for both simulations. Identification accuracy of buses in the RL model was reduced to 75%, with the remaining 25% of objects regarded as background objects. It can be noted from the results in Table 4 that the buses experience much shorter waiting times at the intersection, even if there exists some confusion in the classification model.

Table 4: Mean Accumulated Waiting Time of Priority Simulation

Vehicle Class	Mean Acc Waiting Time (perfect) [s]	Mean Acc Waiting Time (25% bus errors) [s]
Bus	2.3578	3.0412
Passenger Vehicle	39.0824	56.4368

Scenario 4: Multiple intersection FT vs RL with OD errors

The RL model is given the same practical performance as measured from the OD system (70% OD accuracy and maximum absolute error of 50%), to provide the most realistic results for future practical implementation. The RL model improves over time when analysing the accumulated waiting time per vehicle of both intersections over different time steps. Furthermore, the performance of the FT method stays constant and thus the accumulated waiting time of the vehicles starts increasing as time progresses and queue lengths increase. The RL model performs best at time steps past 30K as seen from Table 5 with only 5.3s mean accumulated waiting time per vehicle over a maximum of two intersections.

Table 5: RL and FT mean waiting times per vehicle at various steps

Steps	0-500	0-2500	0-5000	0-50000	30000-50000
RL	28.6099 s	29.0514 s	18.9706 s	7.0046 s	5.2929 s
FT	24.8802 s	29.0514 s	32.4882 s	45.3423 s	52.5222 s

6. CONCLUSIONS AND FUTURE RESEARCH

6.1 Conclusions

The simulations using SUMO show that the traffic density estimation system designed and implemented through this work has sufficient accuracy to allow a reinforcement learning strategy for traffic light control to outperform a fixed cycle time approach. Furthermore, the RL traffic control methods outperform traditional methods in all traffic density scenarios, including priority regulation based on vehicle type and multiple intersections. The results show that the implementation of the combined TDES and RL traffic control software will ensure a more intelligent solution than what is currently used.

The results indicated that it will be worthwhile to invest in better-positioned cameras that will provide better angles to minimise the severity of occlusions for more accurate detections and allow better accuracy of waiting time calculations. The OD and classification model with the lanes defined for each intersection should also be regularly evaluated and updated to maintain high accuracy throughout operation.

6.2 Future Research

Although the implementation produced satisfactory results, some faults need to be addressed in future work. Firstly, negative learning can be implemented to include objects such as pedestrians, cyclists, trailers, and other objects, so that the model is informed on how to distinguish these objects from the interested traffic vehicles. Furthermore, classes such as emergency vehicles (ambulances or police cars) need to be included in these models, to allow additional priority regulation for such vehicles. The dataset created does not include nighttime data or harsh weather conditions data; it is expected that such cases will lead to very bad predictions and consequently bad traffic control in those scenarios. Additional data from those conditions should be gathered and included in the model's training set. Tests should be conducted to observe whether it will then still outperform traditional methods before practical implementation.

The simulations only used mean accumulated waiting time over all lanes as performance measure for traffic flow. Future work should include adding histograms of the waiting time for each vehicle and the lane in which it drove. This will indicate why and by how much the behaviour of specific certain vehicles influenced the mean accumulated waiting time. In the current simulation multiple intersections operated independently with their own RL agents. Implementing a method that incorporates multi-agent communication with synchronization between intersections will result in further improvements in traffic control over larger networks.

7. REFERENCES

Abouelyazid, M. 2023. "Comparative evaluation of SORT, DeepSORT, and ByteTrack for multiple object tracking in highway videos." *International Journal of Sustainable Infrastructure for Cities and Societies,* 8(11): 42-52.

Asha, CS & Narasimhadhan, AV. 2018. Vehicle Counting for Traffic Management System using YOLO and Correlation Filter. 2018 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT): 1-6.

Chung, NM, Le, HD-A, Nguyen, VA, Nguyen, QQ-V, Nguyen, TD-M, Thai, T-T, Ha SV-U. 2022. I. C. o. C. Vision and W. Pattern Recognition. Multi-Camera Multi-Vehicle Tracking with Domain Generalization and Contextual Constraints. IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW): 3326-3336.

Dilek, E. & Dener, M. 2023. "Computer Vision Applications in Intelligent Transportation Systems: A Survey." Sensors (Basel, Switzerland) 23(6).

Farhadi, A, Girshick, R, Divvala, S & Redmon J. 2016. I. C. o. C. Vision and R. Pattern. You Only Look Once: Unified, Real-Time Object Detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR): 779-788.

Fedorov, A, Nikolskaia K, Ivanov S, Shepelev V & Minbaleev A. 2019. "Traffic flow estimation with data from a video surveillance camera." Journal of Big Data 6(1).

Huang, J, Rathod, V, Sun, C, Zhu, M, Korattikara, A, Fathi, A, Fischer, I, Wojna, Z, Song, Y, Guadrrama, S & Murphy, K, 2017. I. C. o. C. Vision and R. Pattern. Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR): 3296-3297.

Janota, A, Kalus F, Pirnik R, Kafkova J, Kuchar P, Skuba M & Holecko P. 2024. Reinforcement Learning Approach to Adaptive Traffic Signal Control Using SUMO-RL. 25th International Carpathian Control Conference (ICCC): 01-06.

Malose, JM, Chunling, T, Pius, A & Mukatshung, N. 2023. "A Vehicle Density Estimation Traffic Light Control System Using a Two-Dimensional Convolution Neural Network," 5: 1844-1862. Doi: 10.3390/vehicles5040099.

Olorunshola, O, Jemitola, P & Ademuwagun, A. 2023. "Comparative Study of Some Deep Learning Object Detection Algorithms: R-CNN, FAST R-CNN, FASTER R-CNN, SSD, and YOLO." *Nile Journal of Engineering and Applied Science*, (0):1.

Prasad, D, Kapadni, K, Gadpal, A, Visave, M & Sultanpure, K. 2019. HOG, LBP and SVM based Traffic Density Estimation at Intersection. 2019 IEEE Pune Section International Conference (PuneCon): 1-5.

Raj, J, Bahuleyan, H. & Vanajakshi, LD. "Application of Data Mining Techniques for Traffic Density Estimation and Prediction." *Transportation Research Procedia,* 17: 321-330.

Shengli, S, Jianmin, X & Zhong, Q. 2006. Vision-based Density Detection of Traffic Flow in Urban Intersection in the Daytime. 2006 6th International Conference on ITS Telecommunications: 187-189.

Shirazi, MS & Morris, BI. 2015. Vision-based vehicle queue analysis at junctions. 12th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS): 1-6.

Sigaud, O & Buffet, O. 2010. Markov decision processes in artificial intelligence: MDPs, beyond MDPs and applications. London, Hoboken, NJ, ISTE; Wiley.

Sutton, RS & Barto, AG. 2020. Reinforcement learning: An introduction. Cambridge, Massachusetts, The MIT Press.

Umair, M., Farooq, MU, Raza, RH, Chen, Q & Abdulhai, B. 2021. "Efficient Video-based Vehicle Queue Length Estimation using Computer Vision and Deep Learning for an Urban Traffic Scenario." Processes, 9:1786. Doi: 10.3390/pr9101786.