Video-Based Monitoring of Red-Light Traffic Law Violation Using Resource-Constrained Edge Computing

by

Ali Zhakiyev

Submitted to the Department of Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science

at the

NAZARBAYEV UNIVERSITY

April 2024

Author .....
Department of Computer Science
April 16

Certified by .....
Dimitrios Zormpas
Assistant professor
Thesis Supervisor

Accepted by .....
Yelyzaveta Arkhangelsky
Dean, School of Engineering and Digital Sciences

Video-Based Monitoring of Red-Light Traffic Law Violation Using Resource-Constrained Edge Computing

by

Ali Zhakiyev

Submitted to the Department of Computer Science
on April 16, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science

Abstract

The need for remote road traffic monitoring is essential to reduce accident possibilities. It encourages drivers to abide by the traffic laws. Recently, researchers have been focused on automatic traffic monitoring using edge devices, Computer Vision (CV), and Machine Learning (ML) due to the increase in vehicle numbers. However, due to the number of edge devices, the problems of high-cost hardware for cloud-based computations and scalability of the bandwidth are arising. Therefore, this paper proposes a low-cost microprocessor-based traffic monitoring system that will conduct all processing on the edge. The system will be used near traffic lights and detect law violations on red lights. It will make drivers more careful by adding certain consequences which will lead to fewer accidents. The microprocessor is equipped with a camera module and is used to run a video processing algorithm and Convolutional Neural Network (CNN) for law violation detection, and its further classification. The device will be installed on the traffic light pole in Astana, Kazakhstan.

Thesis Supervisor: Dimitrios Zormpas
Title: Assistant professor

Acknowledgments

This study was conducted with the help of NU IoT Lab. We appreciate the provided edge devices and the required modules for them. We are also grateful to the supervisor of this research for providing feedback and help throughout research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The rapid growth in population numbers, combined with a higher quality of life, has resulted in an overall increase in the number of vehicles [1, 2, 3]. This, in turn, leads to a higher risk of traffic accidents and more demand for investment in traffic control. Given the surge in both population and vehicle numbers, the need for advanced traffic monitoring systems has never been greater. According to a study conducted in China, growth in both population and the number of cars influences the number of road traffic accidents. For every 10,000 new vehicles, the number of traffic accident casualties, on average, increased by 21.77 [4]. Globally, the consequences of traffic accidents are widespread, affecting not only individual lives but also the economy and urban infrastructure at large. One of the main causes of the accidents is related to violations of traffic laws. Since it is impossible for humans to constantly monitor the traffic, drivers are prone to omit the traffic laws due to the lack of immediate consequences. Therefore, there is a need in traffic monitoring. In the early days of traffic monitoring, human observers were the primary means of gathering data. As cities grew and vehicle numbers increased, manual counting became impractical, paving the way for rudimentary mechanical counters. Over the years, advancements in electronic sensors and digital imaging have revolutionized traffic surveillance, making real-time data collection and automated analysis the new norm. Despite these technological strides, the fundamental challenge of accurately capturing and processing traffic data without prohibitive costs remains persistent. Due to the recent advances in Internet

of Things (IoT) technologies, researchers have been able to counter this issue by using traffic monitoring systems. According to Khan et al., by enforcing the traffic law via monitoring the traffic, road traffic can be significantly decreased [5]. Consequently, research on road traffic monitoring using Machine Learning (ML) and Computer Vision (CV) is receiving attention in the research community.

Currently, most systems constantly collect traffic data. The data is then sent to a remote server where the computation takes place. While being effective for monitoring the traffic, this approach is not scalable in terms of data transfer. Existing solutions often come with substantial infrastructure costs, involving extensive sensor networks and high-end computational resources. A large amount of data forces installment of high-speed transfer methods which require a significant budget and often challenging to deploy in rapidly growing urban centers with diverse traffic patterns. To attain the required computational power, a cloud center is necessary to receive, store, and analyze all of the data produced by the endpoints [6, 7]. Another often-overlooked gap is the environmental impact of deploying large-scale traffic monitoring systems. The energy requirements for constant data transmission and processing place a non-negligible strain on power resources. Therefore, another approach to processing data is to use the computational power of the edge. Even though the unnecessary data is filtered out, the reduction in data size is not scaled to the number of edge devices in a high-end city. Another method in use involves making all the necessary calculations on the edge device, however, current research in the field often uses high-cost powerful edge devices. Therefore, they are not suitable for real-life scenarios, where a large amount of devices are required.

This research aims to design and evaluate a low-cost, scalable traffic monitoring system that maximizes accuracy and minimizes infrastructure requirements by setting focus on computational power of an edge device and optimized lightweight ML model. For this purpose, a stationary system, located near traffic lights, would be used to capture a short video during the red light to detect and report the violation of traffic laws. The picture of the car that violated the law and its license plate is collected and sent to a remote server for formal identification. The system makes all data processing

using a low-budget, power-efficient device such as the Raspberry Pi 4. The footage is collected using a phone camera. This study takes ideas from the study by Kargar et al. [8] and Omidi et al. [9] and improve the accuracy by using the video as input for more data, using newer ML algorithms, and reducing the overall cost of the system by using low-priced alternatives. The main contributions of this research are:

- The reduction of the required bandwidth for the data transfer of the IoT by transferring the calculation to the edge.
- It proposes a lightweight ML model that is suitable for low-powered edge devices.

Chapter 2

Related work

This chapter reviews data filtering approaches on the edge, as well as lane and object detection algorithms.

2.1 Data filtering

The principal method of addressing the challenge of transferring a large amount of data is to filter the data using edge computing. One such method is proposed by Barthélemy et al. for real-time traffic monitoring using CCTV cameras [2]. For their research, the authors use an Nvidia Jetson TX2 ARM-based edge device with the added Pycom LoPy 4 module for the LoRaWAN communications. The algorithm architecture consists of frame collection, object detection, inter-frame object matching, and object trajectory modification and storage. Specifically, YOLOv3 (You Only Look Once) is used for object detection (more details on YOLO in the Object Detection section), and the Simple Online and Real-time Tracking (SORT) algorithm, which uses the combination of a Kalman filter and the Hungarian algorithm, is used for matching objects between frames and achieving multiple objective tracking. [2] are able to reduce the raw footage to the meta-data and successfully produce tracking data.

The same YOLOv3 system is used by Wan et al. [6]. However, rather than generating meta-data from the entirety of raw data, the authors focus on removing

the unnecessary parts by focusing on the specified key points using edge technology. To achieve that, the authors propose their spatio-temporal interest point algorithm to remove redundant video data and multi-modal features combination to get the desired segments of important data. These segments then go through the optimized YOLOv3 algorithm to further increase the speed and quality of detection. According to the results of their research, their approach performs better compared to other algorithms and was able to achieve a precision value of about 95%.

Another approach, the FilterForward, is the video analytics model that uses previously generated computations from the Deep Neural Network (DNN) as input to the specially created microclassifiers (MC) that are installed on the edge device [7]. These MCs are trained for specific events and can correctly determine the segment where these events occur in the video data. According to the authors [7], their FilterForward model can achieve 6.8 times more efficiency compared to other approaches. However, these results are questionable as the experiments are conducted on a desktop computer. Specifically, using Intel® Core™ i7-6700K CPU and 32 GB of RAM. Authors claim that using only the CPU is enough to simulate the capabilities of the edge device that has both CPU and GPU. While the raw computational power might be comparable with the edge device, the amount of RAM is 4 to 8 times the amount of RAM in the average edge device.

2.2 Lane Detection

Since the problem concerns traffic law enforcement, it is important to establish the borders of the monitored road to correctly identify the law violations. However, lane detection also has its problems and specific cases. Mazrouei et al [10], claim that classical lane detection methods are susceptible to shadow noises and unclear or dirtied road lane markings. Therefore, they propose the Hough Transform (HT) based algorithm that is focused on overcoming such difficulties. Where HT is a feature extraction technique based on voting process on parameter space. Specifically, the visual data is first preprocessed using Region of Interest (ROI) cropping to reduce

Table 2.1: Comparison of studies and the proposed system

Study	**Edge device**	**Target**	**Method**	**Performance**
[2]	Nvidia Jetson TX2	Data filtering	YOLOv3 and SORT	Mean accuracy of 69%
[6]	Not specified	Data filtering	YOLOv3	Precision of 95%
[7]	Intel® Core™ i7-6700K CPU and 32 GB of RAM (edge device simulation)	Data filtering	MobileNet DNN	Around 0.8 on F1 score
[10]	Not on edge	Lane Detection	Hough Transform	2% false positive
[11]	Not on edge	Lane Detection	Radon Transform (generalized HT)	0.089 variance of error
[12]	Not on edge	Lane Detection	CNN	F1 score of 93.54% (based on IoU)
[13]	Not on edge	Lane Detection	CNN	96.75% accuracy
[9]	NVIDIA® Jetson Nano B01 ™	Lane, Object, Plane Detection	Canny edge, YOLO, and HT	93.1% accuracy
[14]	Not on edge	Object Detection	YOLO and Fast YOLO	mAP of 63.4 and 52.7
[15]	Not on edge	Object Detection	YOLOv5 and RegNetY002 (CNN)	average F1 score of 90%
[16]	Not on edge	Object Detection	YOLOv7	89% accuracy
[17]	Not on edge	Object Detection	YOLOv3	4 average accuracy of 98%
[18]	Not on edge	Object Detection	Xception and RNN	overall accuracy of 98.9%
[19]	Not on edge	Object Detection	SMC Faster R-CNN	40% improvement from the generic detector%
Proposed solution	Raspberry Pi 4 Model B	Lane, Object, Plate Detection	YOLOv8	Training accuracy mAP50 0.813 for the lane detection and mAP50 0.958 for plate detection. Confidence of 0.84 for vehicle and 0.81 for plate. Accuracy of 88.8% for plate recognition.

the size of the image to the desired state, and applying the 3x3 wiener filter to reduce random noise. The result, then, goes through the segmentation stage where image thresholding, the conversion of the image to black and white, and the HT are performed. Finally, the morphological operator further enhances the lanes by filtering the lane candidates. According to the results of the experiments, the error rate of the proposed algorithm is only 2%.

Another method based on the HT is proposed by Liu et al. [11]. They claim

that developing a generalized low-complexity HT algorithm poses a great challenge. Therefore, they introduce a new method that is based on the Radon transform, a more generalized version of the HT. In their method, the authors use the dictionary learning method to create an approximation of the Radon transform by taking it as a linear transformation. This allows it to execute a Radon transformation on multiple images in parallel. The results of the experiments based around the method proposed by [11] on the RSSCN7 dataset, suggest that their method is around 4 times faster and more accurate compared to the traditional Radon transformation.

In [12] authors propose another approach based on the Convolutional Neural Networks (CNN) rather than traditional CV methods (e.g. HT). The authors claim that in the context of deep learning, lane detection problems are still relevant due to the need for post-processing, training, and lack of open-source solutions. Therefore, they propose LaneATT, the anchor-based single-stage lane detection model that uses a lightweight backbone CNN. Specifically, their model feeds an image through the backbone CNN to receive features for the anchors which then is combined with the set of the global features. This way, the model can incorporate the data from other lanes to help deal with faulty lanes (e.g. obstructed, dirtied, etc.). Finally, the resulting features go through fully-connected layers to identify the lanes from the input image. The results of the tests suggest that the model is as accurate as the second most accurate model and almost 6 times faster than the most accurate model.

Another deep learning-based lane detection method is proposed in [13]. They argue that the existing lane-detecting deep learning models have limitations as they are required to label and pre-process the input data, unnecessarily predict an abundant amount of points, and are too specific. The suggested Point Instance Network (PINet) is designed to overcome these limitations. Specifically, the model is created with the stacked hourglass method, which is generally used for object detection and pose estimation. The input data is first resized, and then it goes through the series of hourglass-shaped architectures to predict the lane key points. Experiments suggest that the model achieves great performance in invisible conditions and has shortcomings in narrow spaces.

Omidi et al., on the other hand, suggest the hybrid method for lane detection [9]. They use both classical CV methods, the Canny edge detector and the HT, and the deep learning model, YOLOv5. Their method uses the Canny edge detector to identify the edges in the image and YOLOv5 to identify the areas where the road markings are located to form a mask of regions. Then this mask is applied to the edges to find the edges that are a part of the road marks. Finally, the HT is applied on the remaining points to identify the road lanes. This method results in 93.1% accuracy.

2.3 Object Detection

Another important concept in edge-based traffic law enforcement is object detection. It is important to correctly identify the object to accurately judge the violation of the law.

The previously mentioned YOLO model is presented by Redmon et al. [14]. YOLO detects objects by separating them into their bounding boxes. The model separates the image into grids where grid cells are responsible for object detection if the object is in the cell region. Specifically, they predict the objects' bounding box and their respectable confidence scores. The image in the bounding box then goes through the CNN and is inspired by the GoogleNet Model. According to the results, the model is 2.5 times faster than the fastest R-CNN while being more accurate.

Moreover, the YOLO model kept being upgraded. In the study by Kazerouni et al. [15], authors use YOLOv5, fine-tuned for traffic sign detection, for their object detection part of the research. The 5^{th} version of the YOLO is created by other authors by using a more complex architecture structure and different training datasets. This study combines the YOLOv5 model with a lightweight CNN-based model for traffic light detection and color classification. They achieved 80%+ accuracy for each class of objects.

Omidi et al. [9] also use YOLOv5, and their proposed architecture uses the model in all of its stages (Lane detection, vehicle detection, plate detection, and character

recognition). The individual steps of the algorithm show 86%+ accuracy while the accuracy of the whole system (each step being accurate for a specific case) is 70.5%.

Tai et al. [16] use the YOLOv7 and the vehicle-to-everything framework to detect objects and traffic flow. The main purpose of their research is to enhance the existing vehicle-to-everything system, used for detecting traffic violations, by integrating the YOLOv7 and SORT. Results show 89% accuracy for red light violations and 78% accuracy for pedestrian violations. [17], on the other hand, proposes a YOLOv3-based system for license plate recognition. The authors especially focused on the multinational license plate layout and used modified versions of YOLOv3 in each step. They achieved an accuracy of over 97% across the different datasets.

Despite the overwhelming popularity of the YOLO model, other neural network models are still being proposed by researchers. Zhang et al. [18] argue that the performances of the other models drop significantly for plate recognition in constrained conditions as they are trained using controlled environments. Therefore, they propose their model which consists of feature extraction using the Xception network and character decoding using the Recurrent Neural Network (RNN). Overall, their proposed algorithm improves the state-of-the-art results by an average of 1%. [19] use R-CNN (Region-CNN) for object detection in traffic. Specifically, they improve on the existing SMC Faster R-CNN. The results of the experiments show a 40% median increase across all traffic datasets.

Table 1 shows a comparison of the mentioned related works. Current studies that focus on edge device computation rely on powerful and costly edge devices. This study proposes a low-cost solution for edge device-based computation and offers a model that is suitable for low-powered devices.

Chapter 3

Methodology

This research is focused on reducing the data flow bandwidth of the traffic monitoring system by transferring necessary calculations to the edge device. It is used to capture road traffic video and report law violations by analyzing the captured video using modern CV and ML technology. The device used in this study is a low-cost single-board computer equipped with a budget camera. The model is required to be lightweight due to the low-powered nature of the single-board computer used in this study and sufficiently powerful to accurately identify violators.

The overall structure of the proposed solution is similar to [9], that is, data acquisition, lane detection, vehicle detection, and plate detection and recognition as shown in Figure 3-1. The data analysis is entirely done on the edge device, Raspberry Pi 4 Model B, and done using the state-of-the-art model, YOLOv8. The model for data analysis is pre-trained on the desktop computer with Nvidia RTX 4070 GPU with 12 GB VRAM (Video Random Access Memory).

3.1 YOLOv8

The YOLOv8 is another version of the popular object detection system. YOLO, by Redmon et al. [14], was created in 2015. Since then, the model has evolved and has been developed further by various researchers. For this research, the eighth installment of the system, created by Ultralytics [20] was chosen for its ease of access,

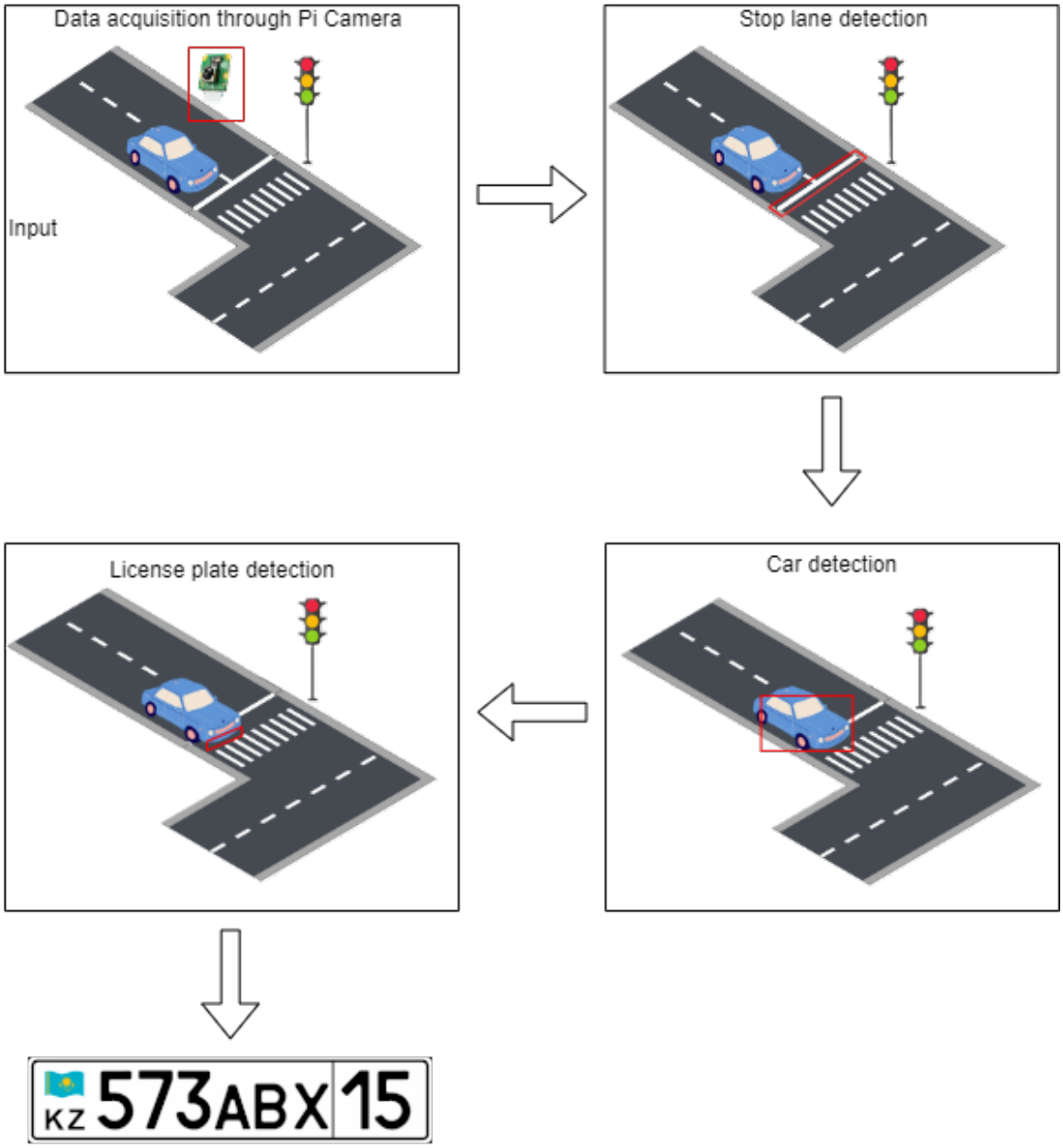

Figure 3-1: Diagram of the proposed system

lightweight version, and the provision of useful tools. Additionally, it provides several models divided by the use case: classification, detection, segmentation, and pose. The entire suite of models offered by Ultralytics supports *track* mode which allows assigning IDs to the specific object to keep them between frames.

The model is built upon YOLOv5, which was modified to include better changes from other versions of the system. The general structure of the model is demonstrated in Figure 3-2. They introduced the segmentation model by utilizing YOLACT (You Only Look At CoefficienTs) [21]. Additionally, they kept different scales of the model from YOLOv5, which are specifically catered for different scenarios. The network of the version was based on the YOLOv7's adaptation of the Efficient Layer Aggregation Network (ELAN). The head module was replaced, changing it to the Anchor-free variant and separating detection and classification models. The data augmentation part, inspired by the training part of YOLOX, was changed to use Mosaic in the later training epochs.

3.2 Data acquisition

The data for the model training is collected from Roboflow[23], a collection of computer vision datasets. For lane and vehicle detection, the research uses a dataset provided by the user *mine* [24]. The dataset features 1936 images of road markings and has 12 distinct labels. For this research, only two of them are used: pedestrian crossing and stop lane. The dataset is divided into training (83%), validation (16%), and test (1%). For the plate detection part, the dataset from the user *PPMG Burgas* [25] is used. It features 675 images of license plates varying in style and form. The dataset is divided into training(70%), validation(20%), and testing(10%). For vehicle detection, a base segmentation-based model provided by YOLOv8 is utilized. Specifically, only car, motorcycle, bus, and truck classes are chosen. The data for the actual processing on the edge device is collected in the form of a 6-second video, captured when triggered by the change of color of the traffic light to red. To assess the accuracy of the proposed methodology, various traffic movement videos were sourced from the

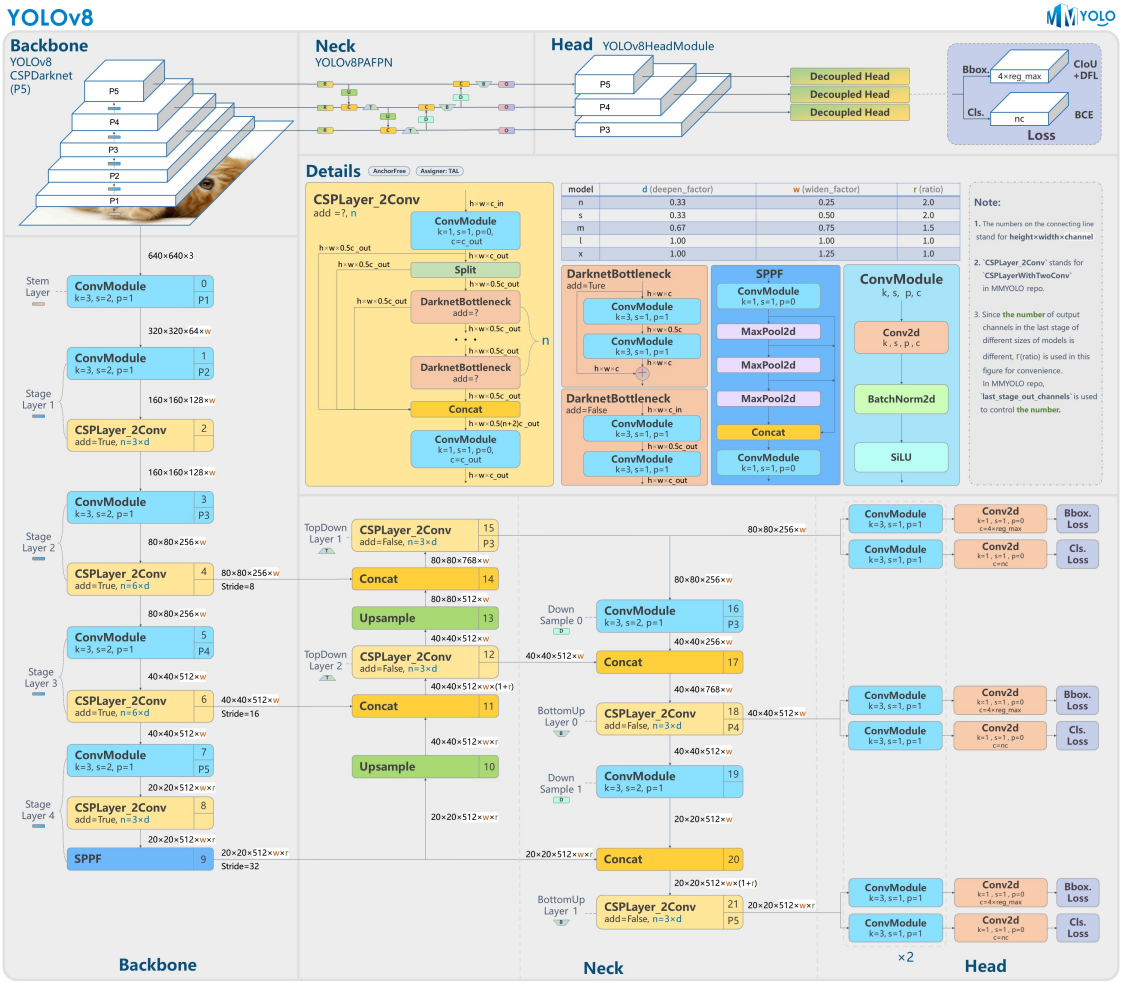

Figure 3-2: The structure of the YOLOv8 [22]

internet. Additionally, the data was manually collected via Pixel 6 phone camera on the streets in Astana.

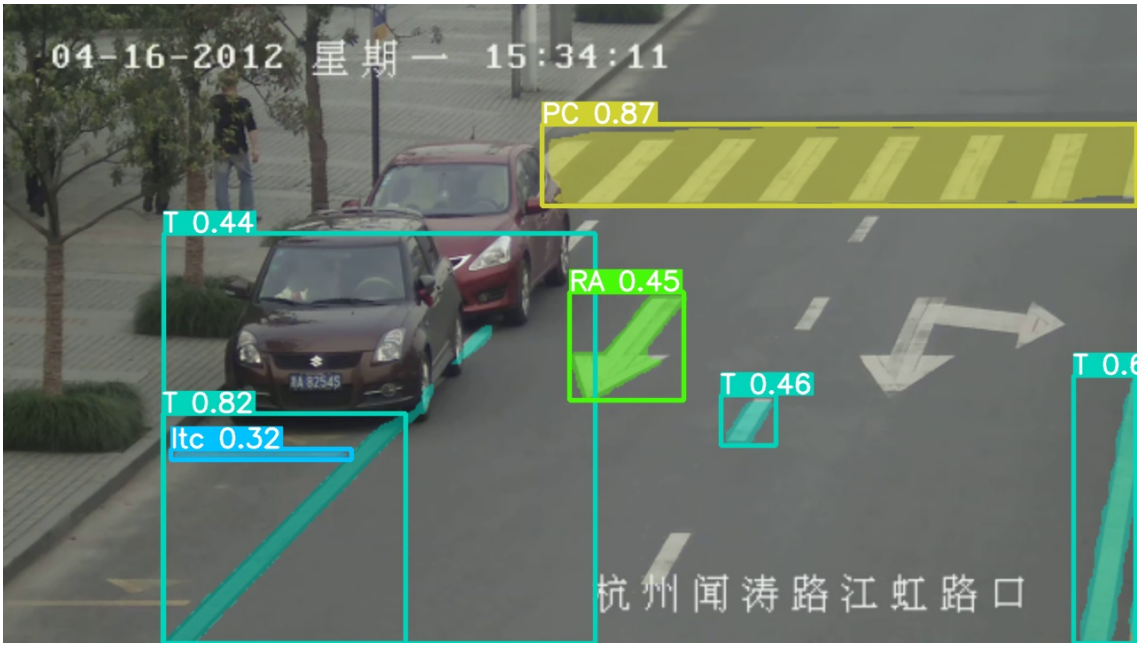

Figure 3-3: Lane detection example

3.3 Lane detection

Since the data acquisition consists of only 6 seconds, it is assumed that vehicles will likely interfere with road lanes. Therefore, the lane detection phase is conducted prior to the commencement of active monitoring, meaning that it will require a separate image where no vehicles are present. As established in the literature, a combination of deep learning techniques and traditional CV methods yields the most accurate results. Specifically, this stage uses an approach similar to [9]. Specifically, the custom re-trained version of YOLOv8 by [20] is used to identify the edges of the road markings. The images in the previously mentioned dataset are resized to 640 by 640 and fed through the YOLOv8n, the lightest available model, for 100 epochs. By monitoring the accuracy scores during training, the best-fitted model is selected to run on the edge device. For the lane detection problem, the re-trained YOLOv8 is provided with a segmentation function to outline the required object. From the result of the

segmentation (e.g. Figure 3-3) it is possible to get the coordinates of the points that make the mask, a set of coordinates of the points surrounding the object, of the object. These road coordinates are then used to identify whether vehicles cross it. For this part, only stop lane and pedestrian lane classes were used.

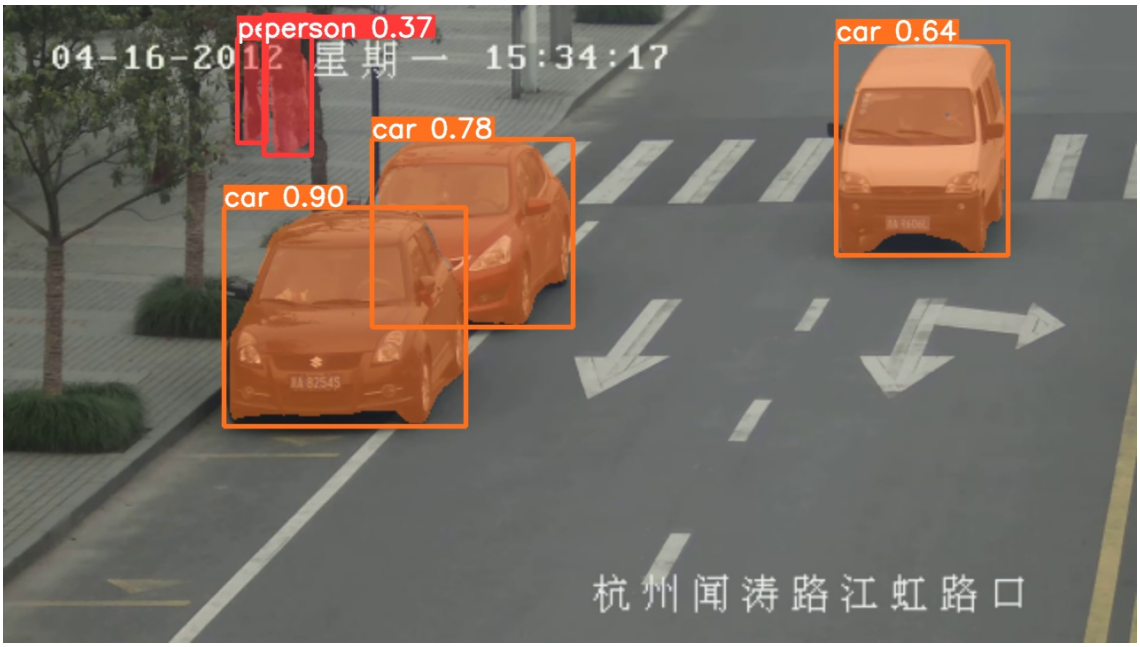

Figure 3-4: Object detection example

3.4 Object detection

After identifying the traffic stop lanes, the subsequent step entails the identification of the vehicles. The YOLOv8 is used again for that task. Specifically, the masks of the vehicles are acquired through the segmentation variant of the base model (e.g. Figure 3-4). These masks are compared with the previously acquired masks of the road markings using the Point Polygon Test function from the Open CV library for Python to identify whether the car violated traffic regulations by encroaching upon the designated road markings.

Figure 3-5: Plate detection example

3.5 License Plate analysis

Finally, for the license plate detection, another specifically trained YOLOv8 model is used. Since this part does not require a precise resulting mask, the less resource-intensive variant of the model is used. The model is re-trained using the plate analysis dataset on the YOLOv8n with images resized to 640x640 pixels and using 100 epochs. The best-fitted model is then chosen to identify the license plate from the image. This part is not always utilized. As shown in Figure 3-1, plate analysis only activates when it is detected that the vehicle runs past previously identified markings. When such a vehicle is detected, only the image within the bounding box of the vehicle is fed through the plate analysis model. This model then identifies the license plate of the vehicle and returns the image within the bounding box, resulting in small images of the plates (e.g. Figure 3-5).

3.6 Testing

The resulting system is tested using a mix of web-sourced and locally-captured videos. The videos were divided into 6-second fragments to simulate the scenario of capturing a short video after the traffic light showed red. Since the system was able to identify the lane crossing in all cases, accuracy is determined by the confidence scores for the object. Additionally, the time required for the system to analyze the video is captured for all test cases to determine the system's efficiency and whether it is suitable for real-world situations.

3.7 Limitations

This study has several limitations due to the equipment failure. The original Raspberry Pi had a failure which removed the possibility of collecting the data on the Pi Camera. Instead, the real-life data was captured using the camera of the smartphone. Additionally, due to the after-winter erosion, the road markings are not visible in Astana at this time of the year. Therefore, the captured data is handled differently. Specifically, the videos were digitally altered to simulate the presence of the pedestrian crossing. Finally, due to the low-power nature of the edge device, the system does not track objects, meaning it is possible to get duplicate recognition of the same vehicle between frames.

Chapter 4

Results

4.1 Training

As mentioned before, the YOLOv8n (smaller version of YOLOv8) is re-trained on a custom dataset for lane detection and plate recognition. The training accuracy, specifically, mean average precision with a threshold of 0.50 (mAP50) for the lane detection is overall 0.834 for the bounding box and 0.813 for the coordinate mask as shown in Figure 4-1.

Table 4.1: Accuracy of the plate detection model (P - precision, R - recall)

Class	**Images**	**P (box)**	**R (box)**	**mAP50 (box)**	**mAP50-95 (box)**	**P (mask)**	**R (mask)**	**mAP50 (mask)**	**mAP50-95 (mask)**
LTDC	305	0.72	0.603	0.608	0.355	0.747	0.587	0.629	0.338
PC	305	0.836	0.875	0.889	0.629	0.848	0.885	0.904	0.612

However, the dataset contains several classes that are not necessary for lane detection, as crossing them is legal. The only classes important for this study are LTDC (stop lane) and PC (pedestrian crossing). As shown in Figure 4-1, PC has a fairly high mAP50 score of 0.889 for the bounding box and 0.904 for the mask. Stop lane class, on the other hand, shows less accurate results of 0.603 on the bounding box and 0.587 on the mask. Since the low score could interfere with the accuracy of the system as a whole, the testing part is mostly done on the pedestrian crossing cases. The plate detection dataset, on the other hand, has only one class. The training

Table 4.2: Accuracy of the plate detection model (P - precision, R - recall)

Class	**Images**	**P (box)**	**R (box)**	**mAP50 (box)**	**mAP50-95 (box)**
License plate	134	0.938	0.929	0.958	0.682

shows an accuracy of 0.929 for r-squared, 0.958 for mAP50, and 0.682 for mAP50-95 (Figure 4-2). Since the plate detection dataset is trained on the recognition model, it only has a bounding box accuracy.

4.2 Efficiency

The system is tested by feeding short, 6-second videos of the traffic through the system to determine the confidence scores of the objects found. Due to a high accuracy of the models, the system captured all of the cases of law violation of crossing a pedestrian crossing on red-light. The cases are simulated, and the data was captured on green light since no real violation was captured during data acquisition otherwise.

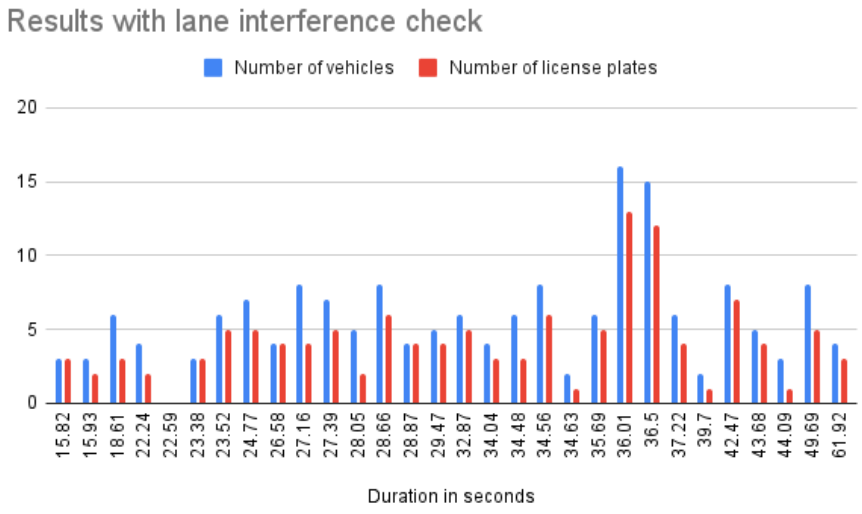

Figure 4-1: Duration of the analysis with lane interference check

The first important metric is the efficiency of the model. That is, how fast it is capable of analyzing the 6-second videos. It is important to finish the analysis of the

video before the next red light to fully operate autonomously. Otherwise, overhead would build up making the system obsolete. Overall, the system spends 0.55-1.6 seconds per object of interest in a frame. Given that every 20^{th} frame analyzed in a 6-second video, the system takes 12 seconds to analyze the video if no violation occurs. On average, the system spends 32.02 seconds for 5.73 vehicles and 4.17 license plates with lane detection, where only the plates from the vehicles that crossed the lane are collected as shown in Figure 4-3. The main problem of vehicle detection in this approach is the presence of duplicates. Due to the light weight of the model, it does not keep track of the objects between frames, therefore, if the vehicle is still crossing the lane in another frame, the still will collect its information again. Of over 172 detected vehicles, 69 detections are duplicates of previously registered vehicles. The plate detection, on the other hand, also contained false detections. Out of 125 found plates, 23 were duplicates and 14 were false detections. This results in an accuracy of 88.8% with duplicates or 86.3% with unique plates.

For the 172 detected vehicles, there are only 111 correctly identified license plates. Further analysis of the detected vehicles shows that missing license plates were obscured by other vehicles, dirt, or not visible due to the camera angle. These points can be easily avoided with properly installed cameras and appropriate red-light traffic.

4.3 Confidence

Another important metric is the confidence of the system in the coordinate masks and bounding boxes of the objects. On average, The vehicles are detected with a confidence of 0.84 across the 172 detected vehicles. On the other hand, the system shows a confidence score of 0.81 for the correctly recognized license plates. Moreover, the system's confidence in falsely detected license plates is 0.5 on average. This implies that it is possible to increase the accuracy of the license plate detection by increasing the confidence metric filter. The result is expected to be higher since the plate recognition is trained on the less precise detection model rather than the segmentation.

Chapter 5

Conclusion

This study suggests a system for autonomous monitoring of red-light traffic and violation analysis. The system consists of a low-cost edge device and a lightweight ML and CV-based model for on-edge analysis. The Raspberry Pi 4 Model B is chosen as a proof-of-concept device with YOLOv8n as a base model for the system. The architecture is capable of analyzing short 6-second videos to determine the law violation and provide license plate information in a picture format, reducing the bandwidth of a traffic monitoring system. The average time for analysis is 32.02 seconds for the system where every 20^{th} frame is used. The low cost of a device makes it affordable to install in large cities with many traffic lights. The proposed system was tested using videos from both the web and real-life scenarios to determine the accuracy of the system. The results of the system suggest an average confidence score of 0.84 for vehicle segmentation and 0.81 for plate recognition with near-perfect accuracy for vehicles and 88,8% accuracy for plate detection. However, the low-power nature of the edge devices does not ensure object tracking, resulting in a high duplicate recognition rate.

The system is easily expandable as technology progresses. The model could be improved for recognition by using the newly created YOLOv9. A lighter and more efficient model might allow the inclusion of a tracking feature to ensure that a vehicle is not detected several times. Another improvement could be done in license plate recognition. The current model provides an image of license plates. This could

be expanded by introducing character recognition and further reducing output by returning results in a string format.

Bibliography

[1] C. Chen, B. Liu, S. Wan, P. Qiao, and Q. Pei, "An edge traffic flow detection scheme based on deep learning in an intelligent transportation system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1840–1852, 2020.

[2] J. Barthélemy, N. Verstaevel, H. Forehead, and P. Perez, "Edge-computing video analytics for real-time traffic monitoring in a smart city," *Sensors*, vol. 19, no. 9, p. 2048, 2019.

[3] A. Frank, Y. S. K. Al Aamri, and A. Zayegh, "Iot based smart traffic density control using image processing," in *2019 4th MEC International Conference on Big Data and Smart City (ICBDSC)*, pp. 1–4, IEEE, 2019.

[4] L.-L. Sun, D. Liu, T. Chen, and M.-T. He, "Road traffic safety: An analysis of the cross-effects of economic, road and population factors," *Chinese journal of traumatology*, vol. 22, no. 05, pp. 290–295, 2019.

[5] N. A. Khan, N. Jhanjhi, S. N. Brohi, R. S. A. Usmani, and A. Nayyar, "Smart traffic monitoring system using unmanned aerial vehicles (uavs)," *Computer Communications*, vol. 157, pp. 434–443, 2020.

[6] S. Wan, S. Ding, and C. Chen, "Edge computing enabled video segmentation for real-time traffic monitoring in internet of vehicles," *Pattern Recognition*, vol. 121, p. 108146, 2022.

[7] C. Canel, T. Kim, G. Zhou, C. Li, H. Lim, D. G. Andersen, M. Kaminsky, and S. Dulloor, "Scaling video analytics on constrained edge nodes," *Proceedings of Machine Learning and Systems*, vol. 1, pp. 406–417, 2019.

[8] A. Kargar, M. P. Wilk, D. Zorbas, M. T. Gaffney, and B. Q'Flynn, "A novel resource-constrained insect monitoring system based on machine vision with edge ai," in *2022 IEEE 5th International Conference on Image Processing Applications and Systems (IPAS)*, pp. 1–6, IEEE, 2022.

[9] A. Omidi, A. Heydarian, A. Mohammadshahi, B. A. Beirami, and F. Haddadi, "An embedded deep learning-based package for traffic law enforcement," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 262–271, 2021.

[10] S. F. Mazrouei and P. Hossein, "A robust and real-time road line extraction algorithm using hough transform in intelligent transportation system application," in *Computer Science and Automation Engineering (CSAE), 2012 IEEE International Conference on*, pp. 256–260.

[11] W. Liu, Z. Zhang, S. Li, and D. Tao, "Road detection by using a generalized hough transform," *Remote Sensing*, vol. 9, no. 6, p. 590, 2017.

[12] L. Tabelini, R. Berriel, T. M. Paixao, C. Badue, A. F. De Souza, and T. Oliveira-Santos, "Keep your eyes on the lane: Real-time attention-guided lane detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 294–302, 2021.

[13] Y. Ko, Y. Lee, S. Azam, F. Munir, M. Jeon, and W. Pedrycz, "Key points estimation and point instance segmentation approach for lane detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 8949–8958, 2021.

[14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[15] A. Kazerouni, A. Heydarian, M. Soltany, A. Mohammadshahi, A. Omidi, and S. Ebadollahi, "An intelligent modular real-time vision-based system for environment perception," *arXiv preprint arXiv:2303.16710*, 2023.

[16] W.-T. Tai, C.-H. Pan, V.-K. Pham, Y.-Y. Lin, and C.-C. Sun, "Vehicles of everything combined yolov7 with traffic enforcement camera on the roadside system," in *2023 International Conference on Consumer Electronics-Taiwan (ICCE-Taiwan)*, pp. 81–82, IEEE, 2023.

[17] C. Henry, S. Y. Ahn, and S.-W. Lee, "Multinational license plate recognition using generalized character sequence detection," *IEEE Access*, vol. 8, pp. 35185–35199, 2020.

[18] L. Zhang, P. Wang, H. Li, Z. Li, C. Shen, and Y. Zhang, "A robust attentional framework for license plate recognition in the wild," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 11, pp. 6967–6976, 2020.

[19] A. Mhalla, T. Chateau, S. Gazzah, and N. E. B. Amara, "An embedded computer-vision system for multi-object detection in traffic surveillance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 11, pp. 4006–4018, 2018.

[20] G. Jocher, A. Chaurasia, and J. Qiu, "YOLO by Ultralytics," Jan. 2023.

[21] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "Yolact: Real-time instance segmentation," 2019.

[22] OpenMMLab, "MMYOLO," Aug. 2023.

[23] "Roboflow." https://universe.roboflow.com/. Accessed: 2024-01-15.

[24] mine, "final-results-5-6 dataset." https://universe.roboflow.com/mine-ch87n/final-results-5-6 , sep 2023. visited on 2024-03-30.

[25] P. Burgas, "Alpr yolov8 dataset." https://universe.roboflow.com/ppmg-burgas/alpr-yolov8 , aug 2023. visited on 2024-03-13.