2021 15th International Conference on Telecommunication Systems, Services, and Applications (TSSA) | 978-1-6654-2892-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/TSSA52866.2021.9768231

Vehicle Traffic Volume Counting in CCTV Video with YOLO Algorithm and Road HSV Color Model-based Segmentation System Development

Abel Stanley
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Indonesia
kumaken132@gmail.com

Rinaldi Munir
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Indonesia
rinaldi@informatika.org

***Abstract*—Traffic congestion is a significant problem in developing countries. One viable solution is a Smart Traffic Light System which utilizes artificial intelligence to adapt light configuration to actual traffic condition in real time. To adapt properly, the system would need traffic density information. We propose a vehicle counting system with neural networks to calculate vehicle volume in traffic roads. In the proposed system, a vehicle is detected with YOLO (You Only Look Once), the state-of-the-art of neural network-based object detection algorithm. The model's performance can be improved with the extraction of RoI (Region of Interest), which is traffic roads. RoI extraction is implemented with HSV color model-based segmentation. Vehicle detection is followed by vehicle tracking and counting. Three tracking algorithms are experimented with the system: KCF (Kernelized Correlation Filter), CSRT (Channel and Spatial Reliability Tracking), and MOSSE (Minimum Output Sum of Squared Error). Vehicle counting is implemented in two methods: incremental or actual. A graphical user interface is developed to provide easy access to system configurations. The result reveals that the best system configuration in terms of accuracy while capable of running in real-time for CCTV recordings is YOLOv4 (608x608) with KCF tracker and RoI Extraction.**

Keywords—artificial intelligence, HSV color model-based segmentation, region of interest extraction, vehicle counting, YOLO

I. INTRODUCTION

Traffic congestion is a significant problem in urban areas in developing countries. Congestion results in time loss, higher fuel consumption, and monetary losses [1]. The capital city of Indonesia, Jakarta, is one of the cities in a developing country that suffers the most from traffic jams. According to data released by the UK's traffic congestion monitoring agency, the TomTom Traffic Index, Jakarta is ranked 10th out of 416 countries with a 53 percent congestion rate during 2019 [2]. According to World Bank calculations, the results of which were submitted by the Minister of Transportation, Budi Karya Sumadi, losses caused by traffic jams in the Indonesian capital exceeded Rp 65 trillion per year [3].

A YOLO (You-Only-Look-Once)-based object detection system is able to analyze CCTV recordings and predict vehicle volume on a route. This vehicle volume can then be used to calculate the path density value which is one of the variables affecting how Smart Traffic Light System allocates time for traffic lights.

Several studies have been carried out regarding vehicle detection in traffic. Four of them are by Kurniawan et al. (2018)[4], Bhuptani et al. (2019)[5], and Asha et al (2018)[6]. The proposed solutions from the studies stated utilized deep neural networks for object detection problem. Some of them also implemented tracking algorithms to maintain detection and identification across video frames. Nevertheless, there are still a few shortcomings from those studies such as counting parked vehicles in traffic density calculation, capable of only calculating the vehicle volume on one lane, and failure to achieve real-time speed to accommodate CCTV cameras. This paper proposes a more robust vehicle counting system that attempts to address each shortcoming previously stated.

II. RELATED WORKS

A. Object Detection

You only look once (YOLO) is state-of-the-art for object detection in real time. Compared to previous research on object detection techniques using classifiers, YOLO adopts a technique that views object detection problems as regression to spatially separated bounding boxes and their associated class probabilities. The original YOLO architecture uses a single neural network so that the bounding box and class predictions are generated in a single pass. Consequently, the YOLO model can be optimized end-to-end directly [7].

Since the first iteration of YOLO, many notable improvements have been made to it by various researchers. The latest improvement of the original YOLO in the name of YOLOv4 has been introduced by Alexey Bochkovskiy et al. [8] The implementation of a new architecture as the backbone of YOLO and modifications made to the neck of YOLO have improved the mean average precision (mAP) of the network by 10% and the number of FPS (Frame per second) by 12%. In addition, YOLOv4 has made the training process of YOLO easier for single GPU systems.

B. Vehicle Counting and Traffic Density Estimation

Kurniawan et al. implemented a simple, basic CNN (Convolutional Neural Network) model to detect traffic density level in Indonesia from manually labeled CCTV dataset and produced an average accuracy of 89.50%. The preprocessing involves converting the input image into 100x100 single channel or grayscale images. Exact vehicle counting is not performed in the project. Instead, the output of the model is a binary classification of whether the image represents a jammed traffic condition or not.

Bhuptani et al. has implemented a system that uses CNN to perform image transformation and is followed by YOLO for traffic density estimation. The result of the system is the counting result classified into 5 vehicle classes, namely car, bus, truck, motorbike, and bicycle. In this project, YOLO is performed after the image size is reduced to a significantly smaller 19x19 image by the CNN, which makes the classification process computationally less expensive and faster. However, this project is unable to count vehicles across frames in videos, which would require tracking algorithms. The paper also stated that the project is unable to exclude parked vehicles, which should not be counted in traffic density calculation.

Asha et al. (2018) has implemented a traffic management system by combining an object detection unit and a correlation filter-based tracker to process traffic data. The object detection unit is built using the state-of-the-art YOLO algorithm and the correlation filter is assisted by scale estimation for vehicle tracking. In the vehicle counting process, the proposed system observes several states to produce a more accurate counting result, namely track state, detect state, terminate state, and target lost state. The solution proposed is stated to only able to count on a single lane. The system fails to count traffic density on each lane separately if there are multiple lanes on the frame.

To overcome the limitations of previous studies, a more robust vehicle counting system that covers every shortcoming previously mentioned is proposed. The proposed system is tailored for practical field use-cases, providing an end-to-end solution to vehicle counting in a video which consists of an object detector and tracker, RoI extraction module, and a graphical user interface to provide better accessibility.

III. PROPOSED METHODOLOGY

The proposed vehicle counting system comprised of a Region of Interest (in this case, traffic roads) extraction module with HSV color model segmentation, a YOLOv4 object detector trained to recognized 2 and 4-wheeled vehicles, an object tracking module in-between detection intervals to reduce average computational load, a vehicle counting module which supports multiple lanes counting, and a graphical user interface (GUI) to provide easy access to every system component configuration. With the assumption that no vehicles are parked in an active traffic road, RoI extraction is sufficient in excluding parked vehicles from the counting result. The flow process of the proposed vehicle counting system is shown in Figure 1.

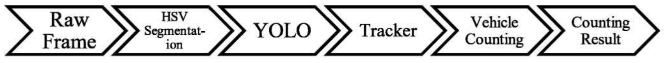


Figure 1. Proposed Vehicle Counting System Flow Process

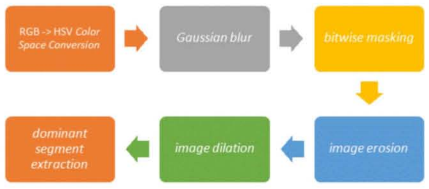
A. Region of Interest Extraction Module

RoI Extraction is performed with HSV color model segmentation. The module uses hue and saturation values most commonly found in roads. HSV color model is chosen because it separates color information (chroma) from intensity or lightning (luma), allowing us to apply segmentation threshold using only hue and saturation. In theory, segmentation done in this way will provide a more robust result regardless of lightning changes in the value channel compared to other color models such as RGB.

Figure 2 illustrates the process pipeline that happens in this module. The preprocess sequence is as follows: first, the raw image in the RGB color space will be first converted to the HSV color space. Then, the image is smoothed with Gaussian blur technique to reduce image detail and Gaussian noise of the image. The kernel size of Gaussian blur operation is chosen following a heuristic. The rule of thumb on determining optimal kernel size of a Gaussian filter is to choose 3 times standard deviation (sigma value) of the image rounded to the nearest odd integer in each direction [9].

After the image is preprocessed, HSV color model-based segmentation is performed. The image is masked using hue and saturation minimum and maximum threshold parameters to extract RoI, which is traffic roads. This results in binary colored mask, white represents RoI pixels and black represents not RoI pixels. Finally, the mask is subjected to post-process sequence to improve segmentation result. Next, image erosion and dilation are performed on the mask. Image erosion calculates local minimum value of an area of the kernel. This causes the white areas (RoI) of the masked image get thinner, while the black areas (not RoI) get wider. Image dilation process calculates the maximal pixel value overlapped by the kernel and replace the image pixel in the anchor point position with the maximal value. This causes white regions within the masked image to "grow". Image erosion removes noise in the mask but downscales the mask. Image dilation upscales the mask back to its original size.

The final step in the post-process sequence is to perform dominant segment extraction. The system finds the contours of the mask and selects the segment with the largest contour area. This step effectively removes every remaining noise in the mask.



RGB -> HSV Color Space Conversion
Gaussian blur
bitwise masking
image erosion
image dilation
dominant segment extraction

Figure 2. Region of Interest Extraction Flow Process

B. Vehicle Detection Model

YOLOv4 object detection model is used to detect vehicles in video frames. Multiple types of YOLOv4 model are experimented with. The models implemented in the system with their configuration discrepancies are shown in Table 1.

Each YOLOv4 model is trained with Pascal Visual Object Classes (VOC) dataset. The dataset used is the aggregation of

Pascal VOC 2007 and 2012. The dataset is also filtered to only include sets containing 4 different classes of vehicles: car, bus/truck, motorbike, and bicycle.

TABLE 1. YOLOV4 MODELS TRAINING CONFIGURATION

Configuration	***YOLOv4 Tiny***	***YOLOv4***	***YOLOv4 Heavier***
Image size	416x416	608x608	832x832
Batch size	64	64	64
Subdivisions	16	32	64
Layers	38	162	162
Base model	yolov4.conv.137	yolov4.conv.137	yolov4.conv.137

Note: Other configurations not listed in the table are left as the default configuration

C. *Vehicle Tracker*

A Tracker is used in tandem with the detection process. Vehicle bounding boxes' location and trajectory given by the detection process are considered and their position in the next frame is predicted by the tracker. Detection process is performed in frame intervals while tracking process is performed in-between detections. In this way, the system is able to perform detection process more seldomly while still maintaining vehicle detections with an object tracker. Thus, computational load done per frame can be reduced, enabling the vehicle detection system to reach a higher FPS.

Tracking algorithm also helps by producing better result in cases where detection fails to, such as when occlusion happens, which is a prominent problem in traffic scene as vehicles are bound to overlap with each other. Three tracking algorithms are experimented with the system: Kernelized Correlation Filter (KCF), Channel and Spatial Reliability Tracking (CSRT), and Minimum Output Sum of Squared Error (MOSSE). Each tracker's performance is measured and compared to determine which tracker is capable of achieving the highest accuracy and is the fastest.

D. *Vehicle Counting Module*

The system offers two vehicle counting methods: incremental and actual. The incremental counting method can incorporate a counting line drawn by the operator and count each vehicle that crosses it. Whenever a vehicle crossed a line, that vehicle will be given the line ID which it crossed to prevent duplicate counting. The module accepts multiple counting lines and perform counting process on each line separately, which makes it possible to perform vehicle counting on more than one lane. Vehicle counting result is accumulated over time. The incremental counting method is best suited for straight roads far from intersections or roads that have several lanes. Figure 3 illustrates the system performing incremental counting method.

The actual counting method performs calculations without any counting line. It outputs the result of the vehicle counting process of the whole scene. The counting tally is deducted every time a vehicle leaves the scene. This counting method is proposed in response to incremental counting method's inability to count vehicles properly in a congested traffic scene. The actual counting method is best suited for intersections, roads near a traffic light, or roads with a single lane.

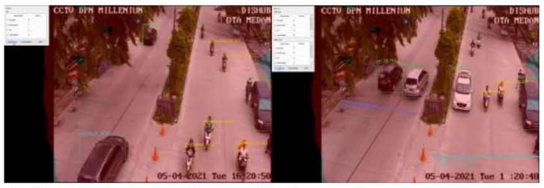


Figure 3. Sample Detection Process (**Left**: actual mode. **Right**: incremental mode with counting lines)

IV. TESTING METHODOLOGY

A. *YOLOv4 Model Test Methodology*

After training, each YOLOv4 model is tested with the Pascal VOC test dataset aggregated from the 2007 and 2012 versions. The main metric used to evaluate model performance is mean average precision (mAP). Mean average precision is a common metric to summarize the precision-recall graph into a single quantifiable value. The average precision of a single query is the mean of the average precision scores over all queries, and the mAP is the mean of average precisions over all queries, which are given by:

$$\begin{aligned} AP(q) &= \frac{1}{N_R}\sum_{n=1}^{N_R} P_q(R_n) \\ mAP(q) &= \frac{1}{|Q|}\sum_{q\in Q} AP(q) \end{aligned} \qquad (1)$$

where R_n is the recall after the nth relevant image is processed and N_R is the total number of relevant items for the query, Q is the set of queries q. The mAP metric is chosen because it considers both precision and recall oriented aspects and is sensitive to the entire ranking [10].

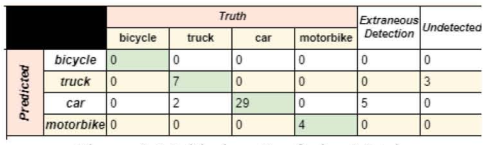
B. *Vehicle Counting Testing Methodology*

The proposed system as a whole is tested on a traffic video never seen before in model training. All tests are conducted with a system supported by Intel(R) Core (TM) i5-8300H CPU @ 2.30GHz, 2304 Mhz, 4 *Core*(s) CPU and NVIDIA GeForce GTX 1050 Ti GPU. The videos used for testing are retrieved from Area Traffic Control System of Medan City in Indonesia. The vehicle counting system is tested on two CCTV videos and each video is tested with both incremental and actual counting methods. The final result is the average of testing result on both videos and counting methods. The ground truth of the test dataset used is obtained by performing manual counting.

First, a multi-class confusion matrix is constructed. Then, the proposed vehicle counting system is run and the confusion matrix is filled as the system counts simultaneously. Every duplicate detection is classified as a false positive and every missed detection is classified as a false negative. Lastly, the values collected in the confusion matrix is used to calculate true positives, true negatives, false positives, and false negatives. Figure 4 shows a sample of multi-class confusion matrix used for testing.

The evaluation metrics considered for testing is Recall, Precision, F-score, Accuracy, False Positive Rate, and

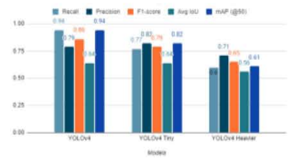
Framerate per Second (FPS). The final result of each metric is averaged with weights relative to their support in ground truth. This is due to vehicle class imbalance in the test videos, containing more cars and trucks than two-wheeled vehicles.

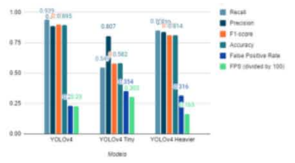


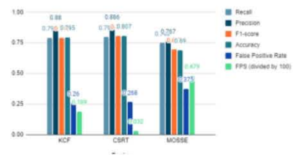
		Truth				Extraneous Detection	Undetected
		bicycle	truck	car	motorbike		
Predicted	bicycle	0	0	0	0	0	0
	truck	0	7	0	0	0	3
	car	0	2	29	0	5	0
	motorbike	0	0	0	4	0	0

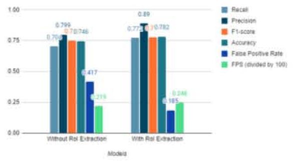
Figure 4. Multi-class Confusion Matrix

V. RESULT AND ANALYSIS









Top Right: Figure 5. YOLOv4 Models Training Result
Top Left: Figure 6. System Performance with Various YOLOv4 Models
Bottom Left: Figure 7. System Performance with Various Tracker
Bottom Right: Figure 8. System Performance with or without RoI Extraction

A. *YOLOv4 Model Training Result Analysis*

Each YOLOv4 experiment model's training result is shown in Figure 5. By comparing each model's precision and recall values, it can be seen that all three models have a balanced ratio of precision and recall. YOLOv4 boasts a considerable gap between 0.94 recall and 0.79 precision. We conclude that YOLOv4 model has a very good detection rate although it can make some false detections sometimes (false positive). YOLOv4 Tiny boasts higher precision compared to recall, though the difference is small. This suggests that YOLOv4 Tiny has a worse detection rate compared to YOLOv4 but more seldom in making false detections. YOLOv4 Heavier, having the lowest F1-score of all three, is lagging behind YOLOv4 and YOLOv4 Tiny in terms of precision and recall balance. This suggests that YOLOv4 Heavier is the comparably worse object detector model.

The mAP value of YOLOv4 Tiny and YOLOv4 models is relatively high (>0.8) while YOLOv4 Heavier mAP is relatively low (0.65). In theory, YOLOv4 Heavier model should have performed better because it processes images at a higher resolution. The contradiction suggests that something has gone wrong in YOLOv4 Heavier model's training process. After further analysis on YOLOv4 Heavier training process, two possible causes are considered: smaller mini-batch size causing the model to be stuck in local optimum [11] and unsuitable anchors settings for given input image resolution.

B. *System Performance based on YOLOv4 Model Analysis*

The proposed system's performance with each YOLOv4 models is shown in **Top Left:** Figure 6. According to the results, the order from the best to the worst model in terms of F1-score and accuracy is YOLOv4, YOLOv4 Heavier, and YOLOv4 Tiny. Contrary to the model training result, YOLOv4 Heavier gave better results than YOLOv4 Tiny.

The cause of the disparity of the result lies on the dataset used during testing. Models are trained with a high-resolution dataset. During testing, the model must perform on CCTV footage with a comparatively lower resolution. This causes YOLOv4 Tiny model, which processes images at 416x416 resolution, to fail in detecting smaller and narrower vehicles properly, such as motorcycles or bicycles. As a result, YOLOv4 Heavier with its lower test mAP value is able to provide better result compared to YOLOv4 Tiny.

C. *System Performance based on Tracker*

The proposed system's performance with each tracking algorithm is shown in Figure 7. According to the results, the order from the best to the worst tracker in terms of F1-score and accuracy is CSRT, KCF, and MOSSE. In terms of speed, the order from the fastest to the slowest tracker is: MOSSE > KCF > CSRT. The FPS difference between MOSSE and CSRT or KCF is significant. The difference between the false positive rate of MOSSE and KCF or CSRT is also quite significant. KCF and CSRT produce a very similar accuracy and F1 score.

D. *System Performance with or without Region of Interest Extraction*

The proposed system's performance with or without RoI extraction is shown in Figure 8. According to the results, it can be concluded that the application of RoI extraction can slightly improve recall, precision, F1-score, and accuracy of the system. More importantly, RoI extraction can significantly reduce the false positive rate and increase the system's average FPS by 3.

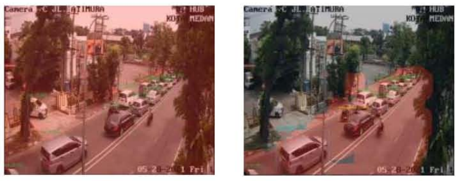


Figure 9. Detection Result without and with Region of Interest Extraction. *(**Left**: without RoI extraction. The false positives: the cars parked on the left side of main road and on far road on top is counted. **Right**: with RoI extraction. The false positives are not counted.)*

RoI extraction is able to improve system performance by preventing some false detections (false positives). Some false positive examples are: parked vehicles, vehicles not on the road of interest, or objects misinterpreted as vehicle such as billboards with car images. By reducing the false positives, the number of objects needed to be tracked decreases. This results in a significant improvement on FPS. By extension, road extraction also excludes parked vehicles as a consequence of excluding roadsides, which is where parked vehicles are most commonly found at. A comparison of detection result with and without RoI extraction is shown in Figure 9.

E. *System Performance based on Configuration Combinations*

The proposed system's performance with each configuration combination is experimented. Every evaluation metric value considered for each configuration combination is laid out in detail by Table 2.

On most configurations, RoI extraction improves the counting result slightly and increases FPS considerably. The only configuration in which RoI extraction becomes detrimental to the performance of the system is with YOLOv4 Tiny and MOSSE tracker. This is because when RoI extraction is performed, when the image frame is masked to a very tight fit with the road of interest, some parts of vehicle might get occluded as a consequence. The small amount of occlusion does not pose a problem for a more accurate YOLOv4 model and tracker, but it is quite problematic for less accurate YOLOv4 Tiny and MOSSE tracker. This causes the system with that configuration to fail in maintaining some detections over the course of a vehicle's movement, resulting in worse accuracy.

The configuration with a good counting accuracy and sufficient FPS for real-time CCTV processing (> 20 fps) is YOLOv4 model, KCF tracker, and with RoI extraction. The system configuration with YOLOv4 model, MOSSE tracker, and with RoI extraction or YOLOv4 Heavier model, MOSSE tracker, and with RoI extraction is not ideal even though it is capable of achieving real-time FPS and good F1-score value. This is because those configurations produce a fairly high false positive rate.

TABLE 2. SYSTEM PERFORMANCE WITH EVERY CONFIGURATION COMBINATION TABULARIZED

Config	*Recall*	*Precision*	*F1-score*	*Accuracy*	*FPR*	*Avg FPS*
Yv4-KCF-RoI	0.944	0.97	0.949	0.963	0.069	21
Yv4-CSRT-RoI	0.956	0.989	0.968	0.981	0.025	3
Yv4-MOSSE-RoI	0.923	0.843	0.857	0.857	0.182	5
Yv4-KCF-NoRoI	0.944	0.879	0.899	0.892	0.313	15
Yv4-CSRT-NoRoI	0.944	0.872	0.895	0.881	0.337	1.5
Yv4-MOSSE-NoRoi	0.923	0.769	0.819	0.798	0.468	47.5
Yv4Tiny-KCF-RoI	0.531	0.866	0.593	0.593	0.274	32.5
Yv4Tiny-CSRT-RoI	0.55	0.884	0.622	0.618	0.246	6.5
Yv4Tiny-MOSSE-RoI	0.47	0.779	0.476	0.477	0.303	57.5
Yv4Tiny-KCF-NoRoi	0.58	0.809	0.616	0.622	0.331	27
Yv4Tiny-CSRT-NoRoi	0.591	0.788	0.633	0.647	0.458	5
Yv4Tiny-MOSSE-NoRoi	0.558	0.713	0.542	0.534	0.511	54.5
Yv4Hvy-KCF-RoI	0.873	0.917	0.866	0.875	0.179	10
Yv4Hvy-CSRT-RoI	0.873	0.93	0.874	0.886	0.161	1.8
Yv4Hvy-MOSSE-RoI	0.829	0.829	0.793	0.784	0.226	40.5
Yv4Hvy-KCF-NoRoi	0.873	0.838	0.826	0.822	0.392	8
Yv4Hvy-CSRT-NoRoi	0.873	0.853	0.835	0.829	0.382	1.3
Yv4Hvy-MOSSE-NoRoi	0.785	0.668	0.696	0.689	0.558	37.5

VI. CONCLUSIONS

This paper proposes and implements a vehicle counting system running on a CCTV video with YOLOv4 object detector, three different tracking algorithms, RoI extraction with HSV color model-based segmentation, and a graphical user interface to operate the system. First, a frame of the video is selected and RoI extraction is performed on it. The resulting image mask is then incorporated into the vehicle detection and tracking module. After the vehicles are recognized and tracked over consecutive frames, they are counted with the vehicle counting algorithm. The system offers incremental and actual counting methods to suit different traffic road situations.

RoI extraction has been shown to improve system performance by reducing the false positive rate and increasing the average FPS of the system. Moreover, RoI extraction allows the system to ignore parked vehicle on the side of the road by only performing detection on traffic roads. The system is also able to perform vehicle counting on two different road lanes using multiple counting lines, each assigned for different lanes.

Finally, it is concluded that the best system configuration in terms of accuracy while capable of supporting real-time FPS for CCTV footage is the system with YOLOv4 (608x608) object detection model, KCF tracker, and with RoI extraction applied. This particular configuration achieves an average of 0.944 recall, 0.97 precision, 0.949 F1-score, 0.963 accuracy, 0.069 false positive rate, and 21 FPS.

REFERENCES

[1] V. Jain, A. Sharma, and L. Subramanian, "Road traffic congestion in the developing world," 2012, doi: 10.1145/2160601.2160616.

[2] TomTom, "Traffic congestion ranking," TomTom Traffic Index, 2019..

[3] Lucky M. Lukman, "Wow, Kerugian Akibat Kemacetan di Jakarta Mencapai Rp 65 Triliun," GalamediaNews.com, 2020. https://galamedia.pikiran-rakyat.com/news/pr-35652610/wow-kerugian-akibat-kemacetan-di-jakarta-mencapai-rp-65-triliun (accessed Nov. 08, 2020).

[4] J. Kurniawan, C. K. Dewa, and Afiahayati, "Traffic Congestion Detection: Learning from CCTV Monitoring Images using Convolutional Neural Network," 2018, doi: 10.1016/j.procs.2018.10.530.

[5] N. Bhuptani, A. Trivedi, and P. Agarwal, "Automating Traffic Signals based on Traffic Density Estimation in Bangalore using YOLO," 2019, doi: 10.1109/ISCON47742.2019.9036213.

[6] C. S. Asha and A. V. Narasimhadhan, "Vehicle Counting for Traffic Management System using YOLO and Correlation Filter," 2018, doi: 10.1109/CONECCT.2018.8482380.

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 2016-Decem, pp. 779–788, 2016, doi: 10.1109/CVPR.2016.91.

[8] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," arXiv. 2020.

[9] Alexei Efros, "Image Filtering and Gaussian Pyramids," CS194: Image Manipulation & Computational Photography, 2017. https://inst.eecs.berkeley.edu/~cs194-26/fa17/Lectures/FilteringGaussianPyramids.pdf (accessed Jun. 01, 2021).

[10] I. Dimitrovski, S. Loskovska, and I. Chorbev, "Efficient content-based image retrieval using support vector machines for feature aggregation," 2010, doi: 10.1007/978-90-481-9112-3-54.

[11] A. Bochkovskiy, "Beta: Using CPU-RAM instead of GPU-VRAM for large Mini_batch=32 - 128," 2019. https://github.com/AlexeyAB/darknet/issues/4386 (accessed Jun. 12, 2021).