S. Tachatanitanont, S. Sricharoenkul

Traffic Congestion Level Identification System Using Deep Learning

***Abstract*—Traffic congestion in Thailand is considered as one of the most serious problems of metropolitan since people who have their own vehicle have to spend lots of time on the road at the rush time of the day and these cause lots of consequences in their lives. According to these problems, knowing the density of the vehicle of the route is important for people to know which route should be avoided at that time. This research presents the use and design of video image processing to analyze the traffic congestion at Samyan-Rama 4 cross intersection. The video dataset is collected from March 8, 2020 to March 21, 2020. This study makes use of a deep learning technology. The selected algorithms ,which consist of "faster_rcnn_inception_v2_coco", "mask_rcnn_inception_v2_coco", "ssd_mobilenet_v1_0.75_depth_coco", and "ssd_mobilenet_v1_ppn_coco", perform to detect and count the vehicles in the frame of video and compare all of 4 algorithms to find the best one according to low error rate, low processing time, and high precision. In order to know the traffic congestion, data collection from the target area has been manually analyzed by humans to know the vehicles patterns. Data has been collected by recording the video at the intersection in a specific period of time which is considered to be peak congestion of the day. In this study, the best algorithm for determining the congestion level is found which is "*faster_rcnn_inception_v2_coco*". This algorithm performs well with 26.60% error rate, 1.60s processing time, and 87.96% precision. From data analyzing by humans, the information which we gain from it consists of the amount of 2 wheels vehicles, amount of 3 & 4 wheels vehicles, duration of green light, and duration of red light. Furthermore, this research also describes the best method, "*faster_rcnn_inception_v2_coco*", of image processing for vehicle detection. The result gives the performance of each method and compares to implement the best one. So any people or organizations who want to determine the traffic congestion level or research about traffic systems and factors that cause the congestion can implement this algorithm for further use.**

Index Terms*—Image processing, deep learning, traffic congestion

I. Introduction

Traffic congestion is one of the biggest problems in Bangkok, the capital city of Thailand. Rising traffic congestion is an inevitable condition in large metropolitan areas. Nowadays, every weekday people have to face a problem like being late at work. If people know how much congestion in each direction, it will be perfect to make them decide to go on that way or not. Knowing traffic congestion levels in each area is important to improve transportation in Bangkok, which route should be adjusted for the traffic flows. So the method that is used to handle this problem must involve technology to maximize the work performance and minimize the used time and efforts.

From the present trend, one of the most up-to-date, acceptable, and comfortable ways to analyze traffic data in Bangkok is video image processing by deep learning using object detection algorithms. Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class in digital images and videos[4]. This technology has several methods for examples: Region-Convolutional Neural Network[5], Fast Region-Convolutional Neural Network[6], Faster Region-Convolutional Neural Network[7], Single Shot Detector[8], and You Only Look Once[9].

In order to reach the goal, traffic data has been collected by recording a video at the rush hour and then analyzing the video to gain the insight information. It is difficult to identify and select the best method of object detection for determining the traffic congestion because many studies are different at many stages, as well as different dataset and test conditions. So, multiple object detection methods are investigated and compared under common conditions to find the best one.

II. Background

According to the current situation in Thailand about traffic congestion, Bangkok is considered as the highest congestion level among all of the regions in Thailand[2]. As Bangkok is the capital and centralized city, almost traffic congestion daily occurs in the downtown area [1]. In order to lack public transportation[1], people tend to use their own vehicles to go anywhere they want rather than using public transportation and these cause a dense vehicle on the roads[1]. According to TomTom reports[3], in 2019, Thailand was 11 world rank from over the whole world of high traffic congestion ,which is 53%. TomTom also shows that the worst congestion of the day occurs at 6-7pm and the worst day is on Friday[3]. Moreover, in one year each person has to lose the time on the road for 8 days and 15 hours on the road due to TomTom report[3]. According to this serious problem, people will lose lots of their time on the road for nothing. To enhance the problem, it must be involved with the use of technology to maximize the performance and minimize the processing time and errors.

With the use of technology, people are able to get to destinations more quickly and more efficiently. For example, Google maps, it uses green, yellow, and red routes to indicate clear, slow-moving, and heavily congested traffic[11]. This function helps people to know the estimated time for getting to

their destinations. Moreover, it provides the several routes with an estimated time of all routes it shows. Google maps applies deep learning techniques to predict the traffic views and recommended routes based on historical data which is sent by embedded sensors in smartphones [11]. So the data, which is continuously sent from embedded sensors, is tremendous and impossible to analyze and store if people who want to do it are general users with 1pc or laptop.

The research paper from Iowa State University, "*Real time traffic congestion detection using images"* [10], uses a deep learning algorithm to detect whether an image shows traffic congestion. This research also uses a set of the images and the persistence check, and then identifies the time of congestion started and when it will end[10]. This research describes all of the steps and the used methodologies[10]. According to requirements of real-time processing and legally accessing various CCTV cameras from various places[10], it requires lots of cost for data collection, data storage, processing power, and processing machines.

According to Tensorflow detection model zoo from Github[15], it provides a collection of detection models pre-trained on the COCO dataset, Kitti dataset, Open image dataset, the AVA v2.1 dataset and the iNatuaralist Species Detection Dataset. All of these models can be useful for out-of-the-box inference if you are interested in categories already in those datasets. They are also useful for initializing your models when training on a novel dataset. All of methods performance for processing the task on Nvidia GeForce GTX TITAN X card are shown in table 3.

From table 1, all of the object detection models and their performance are shown. In order to detect the vehicle in the frames and determine congestion level, four methods are selected to be candidates for this research which consists of "faster_rcnn_inception_v2_coco", "mask_rcnn_inception_v2_coco", "ssd_mobilenet_v1_0.75_depth_coco", and "ssd_mobilenet_v1_ppn_coco" and their performance are shown in table 2.

Although many countries can reduce the traffic congestion by using fantastic technologies, Thailand's traffic congestion level remains on the top list of the worst countries for traffic congestion because Thailand does nothing to solve this problem seriously. This research aims to find the best method of object detection based on accuracy, performance, and processing time. The goal is to reduce and determine the current traffic congestion at that time in Thailand by using an object detection method, but first it should start from the small scale so the focussed area on this study is the Samyan-Rama4 intersection. This intersection is near Chulalongkorn University and any students who want to go to study in the morning or go back home in the evening have to use this route. So if the traffic congestion on this route, Samyan-Rama4 intersection, is enhanced, all of Chulalongkorn's students, staff, and teachers will have a better quality of life and emotion by reducing the time waste on the road.

Table I. List of all object detection models and their performances [15]

Model Name	Speed(ms)	COCO mAP [^1]	Output
ssd_mobilenet_v1_coco	30	21	Boxes
ssd_mobilenet_v1_0.75_depth_coco	26	18	Boxes
ssd_mobilenet_v1_quantized_coco	29	18	Boxes
ssd_mobilenet_v1_0.75_depth_quantized_coco	29	16	Boxes
ssd_mobilenet_v1_ppn_coco	26	20	Boxes
ssd_mobilenet_v1_fpn_coco	56	32	Boxes
ssd_resnet_50_fpn_coco	76	35	Boxes
ssd_mobilenet_v2_coco	31	22	Boxes
ssd_mobilenet_v2_quantized_coco	29	22	Boxes
ssdlite_mobilenet_v2_coco	27	22	Boxes
ssd_inception_v2_coco	42	24	Boxes
faster_rcnn_inception_v2_coco	58	28	Boxes
faster_rcnn_resnet50_coco	89	30	Boxes
faster_rcnn_resnet50_lowproposals_coco	64		Boxes
rfcn_resnet101_coco	92	30	Boxes
faster_rcnn_resnet101_coco	106	32	Boxes
faster_rcnn_resnet101_lowproposals_coco	82		Boxes
faster_rcnn_inception_resnet_v2_atrous_coco	620	37	Boxes
faster_rcnn_inception_resnet_v2_atrous_lowproposals_coco	241		Boxes
faster_rcnn_nas	1833	43	Boxes
faster_rcnn_nas_lowproposals_coco	540		Boxes
mask_rcnn_inception_resnet_v2_atrous_coco	771	36	Masks

mask_rcnn_inception_v2_coco	79	25	Masks
mask_rcnn_resnet101_atrous_coco	470	33	Masks
mask_rcnn_resnet50_atrous_coco	343	29	Masks

Table II. Top performer of object detection models and their performance.

Model Name	Speed (ms)	COCO mAP[^1]
faster_rcnn_inception_v2_coco	58	28
mask_rcnn_inception_v2_coco	79	25
ssd_mobilenet_v1_0.75_depth_coco	26	18
ssd_mobilenet_v1_pnn_coco	26	20

III. Design Description

A. Overview

The system can be used to analyze and identify traffic congestion levels at any area that users are interested in. Basically, the system will analyze an input which is a live video feed from a cctv camera at the interested intersection and at every period of 15 minutes, the system will return an output as traffic congestion level which will later be sent as an input for traffic simulation.

First, there are two important variables that need to be initiated, a timer and a vehicle counter variable. The timer is a python timer which will be used to determine whether the system has reached the 15 minutes threshold or not. As for the vehicle counter, it will be used to count the amount of vehicles that had appeared in the frame which will later be used to identify a traffic congestion level. At the beginning, the system will initiate a timer which is a python timer. Then, the system will start detecting the object in the live video feed. After the system can detect an object in the frame, it will then start to analyze whether that object is a vehicle or not. The model will classify the object as a vehicle if the detected object's class falls into the 4 following classes including car class, motorcycle class, bus class, and truck class. If the detected object was classified as a vehicle, the system will then increment the vehicle counter variable up by one and draw a bounding box around the detected object. If not it will start to detect other objects in the frame. The system will detect and classify all objects that appear in the frame until the python timer reaches the 15 minutes threshold. When the timer has reached its threshold, the vehicle counter variable and the timer variable itself will be reseted to zero.

After the 15 minutes marks, the system will start to compute an average car speed for that 15 minutes period. To compute this value, two things are needed including the total number of vehicles that passed through the intersection and the car headway. The total number of vehicles that passed through the intersection is already stored in the vehicle counter variable which is ready to be used for computing the average car speed in a 15 minutes period. As for the car headway which is defined as a safe distance between cars, the Department of Disaster Prevention and Mitigation, Ministry of Interior, Thailand, recommended that every car should make a headway of at least 60 meters or 0.06 kilometers apart from each other in order to safely stop the vehicle in case of emergency [13]. Hence, this system will compute the average car speed using a headway value of 60. The average car speed, first traffic flow and traffic density will be computed first, then the average car speed will be equal to traffic flow divided by traffic density. The traffic flow is computed by using the vehicle counter divided 15 minutes, then converted that value into a unit of vehicles/hr. For example, if there are 100 vehicles passed through an intersection in 15 minutes, the traffic flow will equal to 400 vehicles/hr. As for the traffic density, it is the reciprocal of headway, so if the headway is 0.06 kilometers, the traffic density will equal to roughly about 16.67 vehicles/km. From this, we can compute the average car speed as 400 divided by 16.67 which is approximately equal to 24 km/hr.

After the system had computed the average car speed, it will then classified whether the computed is in which level of traffic congestion based on the following baseline (higher level indicate more congestion) :

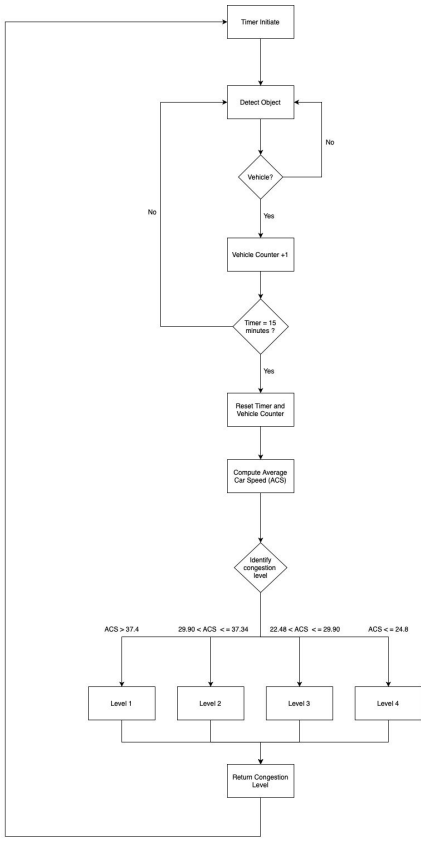
a) Congestion Level 1: average car speed > 37.34km/hr
b) Congestion Level 2 : 29.90 km/hr < average car speed < = 37.34 km/hr
c) Congestion Level 3: 22.48 km/hr < average car speed < = 29.90 km/hr
d) Congestion Level 4: average car speed < = 24.8 km/hr

The above baseline was computed by using the maximum average car speed and minimum average car speed information in 2019 based on the information provided by [14]. This source provided a statistic of average car speed on Bangkok's road at different periods of time. However, it only provided information from 2011 to 2016, so to get the information of 2019 further analysis needed to be done. The analysis is shown in Table III.

Table III. Average Vehicle Speed in Bangkok

Year	Maximum Average Car Speed (km/hr)	Minimum Average Car Speed (km/hr)	Maximum Speed Change (per year)	Minimum Speed Change (per year)
2011	45	30	-	-
2012	46	30	2.22%	0.00%
2013	42.5	27	-7.61%	-10.00%
2014	41	27	-3.53%	0.00%
2015	40	25	-2.44%	-7.41%
2016	40	25	0.00%	0.00%
Average			**-2.27%**	**-3.48%**
2017	39.09	24.13		
2018	38.20	23.29		
2019	**37.34**	**22.48**		

Figure I. System's Flowchart



B. Use

The system in this report can be used for analyzing the traffic on different areas that have a traffic CCTV camera installed based on the interest of the users. To use it, simply connect the system to the traffic cctv camera that was installed on the intersection that is an area of study. The main purpose of this system is to be integrated with the traffic simulation system in order to simulate and determine the best option to travel from one point to another point in Bangkok using a vehicle. The output of the traffic congestion identification system will be used as the input in the traffic simulator, so the simulator can easily identify congestion levels at different interested areas.

Moreover, the proposed system can be used to help ease the traffic congestion problem on Bangkok's road. The system itself can also be used to get the number of vehicles passed through any intersection that has a traffic camera installed. If those cameras are connected to the system, the amount of vehicle can be identified and can be used further in traffic analysis to help find the root cause of the problem in Bangkok's road. This can be done by having a government or any organization that works in the area related to road traffic adopting this system and integrated into the current technology that they are using.

IV. Experimental Setup And Methods

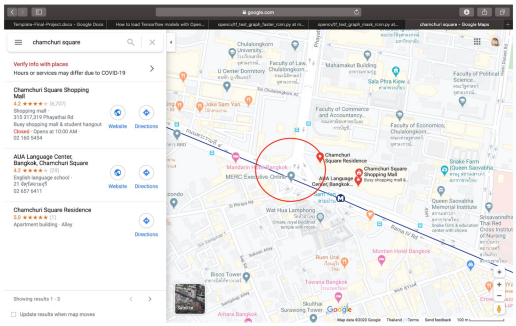
A. Experimental Setup

Since there are many classification models to choose from, a comparison of those models needs to be done first since each model has a different usage and their performance will vary from one scenario to others. For this report, 4 models will be compared with each other to find out which model performs the best based on its speed versus accuracy tradeoff. The 4 models consisted of "*ssd_mobilenet_v1_0.75_depth_coco*", "*ssd_mobilenet_v1_ppn_coco*","*faster_rcnn_inception_v2_coco*", and "*mask_rcnn_inception_v2_coco*" [15]. These 4 models were trained on COCO dataset [16] which is a large image dataset designed for object detection, segmentation, person key points detection, segmentation, and caption generation. To test each model, a python program that uses the opencv library to read the model and use it to detect a vehicle in each video frame is needed to be developed. To run the model using the opencv library, a protobuf file (.pb file) and a proto text graph file (.pbtxt) are needed. However, the model from [15] only provided the protobuf file for each model, so the proto text graph file is needed to be created manually using [17] for SSD and SSD-PPN model, [18] for Faster-RCNN model, and [19] for Mask-RCNN model.

Each model will be tested on the same dataset which consists of images taken from the video records that were recorded at Chamchuri Square intersection from 6pm to 7 pm for 14 days, shown in Figure II. This time period is chosen since it is the peak time of traffic congestion based on [12]. The peak time of traffic congestion is chosen since at this time

vehicles on the road are packed closely to each other which can be used to test the performance of the detector to its limit.

Figure II. Test Data Collection Area



Each test image in the dataset is a frame shot captured from the video records. Based on Table 2, the average duration of red light is at around 157 seconds and the average duration of green light is at around 38 seconds. This means the time gap between two sets of vehicles is at around 195 seconds to 352 seconds which has a median at 273.5 seconds or 4.33 minutes. If one video is 3600 seconds long and each frame is divided in a way that they are 273.5 seconds apart, there will be around 13 frames for 1 video. Hence, the dataset will contain a test data in a total of 182 images.

Table IV. Average Duration of Traffic Light at the Intersection

Day	Total Rounds of Light Signal	Total Green Light Duration	Total Red Light Duration	Average Duration per 1 Green Light	Average Duration per 1 Red Light
1	18	683	2812	37.944	156.222
2	16	569	2341	35.562	146.312
3	21	613	2814	29.19	134
4	19	701	2896	36.895	152.421
5	17	744	2706	43.765	159.176
6	12	744	2809	62	234.083
7	21	610	2886	29.047	137.428
8	21	612	2829	29.143	134.714
9	19	664	2843	34.947	149.632
10	21	609	2865	29	136.429
11	21	611	2855	29.095	134.952
12	13	720	2874	55.385	221.077
13	17	820	2670	48.235	157.059
14	21	615	2886	29.286	137.429
Total					
Average	18.357	665.357	2791.857	37.821	156.495

B. Experimental Methods

To examine the performance of each model, a python program is developed using the opencv library to read the protobuf file and proto text graph file. The program will receive an input as test images and return outputs as the number of vehicles that each model can detect and the processing time. Vehicle in this context includes a motorcycle, 3 wheel vehicle, and 4 wheel vehicle. The overall performance of each model was measured by 7 variables as followed:

a) Processing Time: This variable is measured in second which indicates how long a model took to process a single frame. This variable was measured by the python program itself by using a python time library. First get the current time (start time) before forwarding the image into the model's network, then after the image has reached the output node, get the current time again (end time). The processing time will be the difference between the end time and the start time.

b) Vehicle Detected: This variable is one of the outputs of the python program, the number of vehicles that a model can detect is stored in a python variable called vehicleCount which is set to be zero at the beginning. Each time that the model can identify an object as a vehicle, the vehicleCount will increment by one and when the model cannot further identify more objects, the program will return the final value of this variable in the console.

c) Vehicle in the Frame: This variable is the real amount of all vehicles appearing in the frame which is visible to human's eyes which is obtained by using a human to count all of the vehicles in the frame manually.

d) Error Rate: This variable is a percentage difference between vehicle detected and vehicle in the frame. It was computed by simply taking a result from an absolute value of a difference between vehicle detected and vehicle in the frame, divided by vehicle in the frame, then multiplying that value by 100.

e) TP: This variable is the true positive value which was defined as the number of times that the model was able to identify the object into a correct class which was examined manually using human's eyes. One object cannot have more than one TP.

f) TN: This variable is the true negative value which was defined as the number of times that the model identified an

object to be into an incorrect class. Moreover, if the model identifies an object to be in a certain class more than once, the excess counts were considered as TN. For example, if a car in a frame was identified by the model to be in the vehicle class twice, the first time will be considered as TP, but the second time will be considered as TN.

g) Precision: This variable shows how accurate the model can identify an object to be in a vehicle class in percentage which was computed by dividing TP with the sum of TP and TN, then multiplying the result by 100.

V. Results And Discussion

A. Result

The table V. provided above is the results summary for each model. For the full results of each model, it can be found in Appendix A, B, C, and D. For the data that were used human's eyes to identify such as TP, and vehicle detected, the human's eyes that were used for analysis had a maximum operating time of 3 hours. After 3 hours, the eyes were rested at least 30 minutes before further continuing the analysis in order to avoid any damages to the eyes. Also, this helped decrease the human error in data entry and in the analysis.

Table V. Summary Result of each model

Model Name	Average Error Rate (between actual & detected)	Average Processing Time (seconds)	Average Precision
faster_rcnn_inception_v2_coco	26.60%	1.60	87.96%
mask_rcnn_inception_v2_coco	26.81%	4.30	26.81%
ssd_mobilenet_v1_0.75_depth_coco	81.73%	0.10	98.30%
ssd_mobilenet_v1_ppn_coco	47.91%	0.12	79.39%

Considering the error rate, the third model in table V. performed the worst with the error rate as high as 82%, its error rate is 3 times higher than the model that has the best performance in the aspect of error rate which is the first model in table V. Although that, if judging by the processing time, the third model in the table V. is the best performer with a processing time closely to real time processing. The first model in the table V. also got the highest precision rate with a rate of close to 100%.

B. Discussion

The third and fourth models in table V. got an impressive processing time since they used a single shot detector or SSD as a backbone of the model. Single shot detector is a single stage detector. By comparing this type of detector to a two stage detector like the detector in the R-CNN family, the single stage detector will have a lower processing time compared with the detectors which used a two stage approach since the single stage detector only needs to take one single shot to detect multiple objects within the image. However, this benefit comes with an accuracy tradeoff causing the third and fourth model in the table V. to have a high error rate. As for the first and second model in table V., they used a region-based convolutional neural network (R-CNN) as a backbone in their neural network which means they used a two stage approach to identify and classify an object. The detector that used a two approach will first propose the region of interest (ROI) by using a select search or regional proposal network. After that, it then uses a classifier to process the proposed ROI causing this type of detector to have a much lower error rate than the single stage detector.

As stated in the beginning of this report, the model which will be used for the system needed to have a good performance in terms of speed versus accuracy tradeoff. By considering the results shown in the Result part, in terms of processing time and error rate, the best performers are the first and second models in table V, respectively. First, if comparing the processing time of the first model and the second model in table V., the difference of their processing time is quite low. However, if looking into the difference in their error rate, the difference is much more significant than the processing time's difference. Hence, the third model in table V. should not be used in the traffic congestion identification system since it had too high error rate. Also, the fourth model in table V. should be cut off from the consideration since it had an error rate almost times bigger than the first model in table V.

Judging between the two remaining models, the first and second models in table V., their precision and error rate are close to each other. However for the processing time, the first model can process a single frame about 3 seconds faster than the second one. Since the traffic identification system should be able to process a video frame near real time as much as possible, the first model should be the best choice for the system.

The system can be used by anyone ranging from regular road users to organizations dealing with road traffic. Regular road users can adopt this system to analyze the traffic flow of the road that they use regularly. They can make a video record of traffic flow in the area that they are interested in and run the system to give them the congestion level and amount of vehicles in each period. This information can help them analyze how the traffic changes in a certain period of time and help them be able to plan ahead and optimize their time on the road.

VI. Conclusion

One problem that every road user in Bangkok is currently facing is traffic congestion. Traffic simulation is one way to help ease this problem, this can help road users in

Bangkok which alternative way they can use in order to avoid the traffic. For the simulation to have a high accuracy, the system needs to accurately identify the congestion level in each area in a near real time process. One way to do that is to adopt the image processing technique, using this technique, the system can directly get input images from the CCTV camera, analyze the image and identify the congestion level. Since there are many types of detector to select from, their performance needed to be tested before choosing one to implement into the final system.

According to this study, the traffic patterns have been analyzed and presented in this research. For the analysis part, it consists of 3 main steps: recording the videos on the focussed area, watching the videos and recording the important information, and putting the information from watching the videos to the tables in order to be easy-to-understand. All of the traffic information has been recorded and compared to each day to see patterns. All of the information has been used for model testing and evaluation.

As a result from model testing, the first model, “faster_rcnn_inception_v2_coco”, performs the best performance in every requirement. With 26.60% error rate, 1.60s processing time, and 87.96% precision would be the selected model for the best one according to criteria. So it is obviously cleared that “faster_rcnn_inception_v2_coco” is appropriate for implementing the system.

To conclude, this study ,which is about applying deep learning to detect and count the vehicle in the video, and determine the congestion level in the frame, can find the appropriate model for detecting the vehicle and determine the congestion level of traffic in the video frame base on three criteria, error rate, processing time, and precision. But in order to maximize the system’s performance, it needs some further improvement in the future like how to reduce a processing time, how to reduce an error rate, and how to increase precision.

References

[1] Bangkok Post Public Company Limited, “Understanding Bangkok's traffic woes,” https://www.bangkokpost.com. [Online]. Available: https://www.bangkokpost.com/opinion/opinion/1762349/understanding-bangkoks-traffic-woes. [Accessed: Apr. 9, 2020].

[2] Kubota, “BANGKOK - WORLD'S WORST TRAFFIC JAMS AND THEIR CAUSES,” WHEEL EXTENDED, 31-Aug-1996. [Online]. Available: https://trid.trb.org/view/470865. [Accessed: Apr. 1, 2020].

[3] “Bangkok traffic report: TomTom Traffic Index,” report | TomTom Traffic Index. [Online]. Available: https://www.tomtom.com/en_gb/traffic-index/bangkok-traffic/. [Accessed: Nov. 10, 2019].

[4] S. Dasiopoulou, V. Mezaris, I. Kompatsiaris, V.-K. Papastathis, and M. Strintzis, “Knowledge-assisted semantic video object detection,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 15, no. 10, pp. 1210–1224, 2005.

[5] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation,” 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014.

[6] R. Girshick, “Fast R-CNN,” 2015 IEEE International Conference on Computer Vision (ICCV), 2015.

[7] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137–1149, Jan. 2017.

[8] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “SSD: Single Shot MultiBox Detector,” Computer Vision – ECCV 2016 Lecture Notes in Computer Science, pp. 21–37, 2016.

[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[10] R. A. Somayajula, “Real time traffic congestion detection using images,” Iowa State University Digital Repository. [Online]. Available: https://lib.dr.iastate.edu/creativecomponents/2/. [Accessed: Apr. 5, 2020].

[11] B. Brindle, “How Does Google Maps Predict Traffic?,” HowStuffWorks, 31-Oct-2014. [Online]. Available: https://electronics.howstuffworks.com/how-does-google-maps-predict-traffic.htm. [Accessed: Apr. 1, 2020].

[12] Puntumapon, Kamthorn & Punyararj, Taksin & Pattara-atikom, Wasan. (2009). “Bangkok Road Traffic Characteristic" [Online]. Available: https://www.researchgate.net/publication/279951324_Bangkok_Road_Traffic_Characteristic [Accessed: Nov. 25, 2019].

[13] Department of Disaster Prevention and Mitigation (2017). “Recommendation on Road Safety” [Online]. Available: http://www.disaster.go.th/th/cdetail-11635-news-207-1/ปภ.แนะผู้ขับขี่เรียนรู้หลักปฏิบัติในการขับรถอย่างปลอดภัย...ป้องกันอุบัติเหตุชนท้าย [Accessed: Apr. 14, 2020].

[14] “ความเร็วเฉลี่ยของยานพาหนะใน กทม. และปริมณฑล: Longdo Traffic รายงานสภาพจราจรกรุงเทพและประเทศไทย: Bangkok and Thailand Traffic Information Report ข้อมูลจราจรล่าสุด,” Longdo

Traffic รายงานสภาพจราจรกรุงเทพและประเทศไทย | Bangkok and Thailand Traffic Information Report ข้อมูลจราจรล่าสุด. [Online]. Available: https://traffic.longdo.com/averagespeed. [Accessed: Apr. 5, 2020].

[15] Tensorflow, “tensorflow/models,” GitHub. [Online]. Available: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/. [Accessed: Apr. 10, 2020].

[16] “Common Objects in Context,” COCO. [Online]. Available: http://cocodataset.org/#home. [Accessed: Nov. 20, 2019].

[17] Opencv, “opencv/opencv,” GitHub. [Online]. Available: https://github.com/opencv/opencv/blob/master/samples/dnn/tf_text_graph_ssd.py. [Accessed: Mar. 24, 2020].

[18] Opencv, “opencv/opencv,” GitHub. [Online]. Available: https://github.com/opencv/opencv/blob/master/samples/dnn/tf_text_graph_faster_rcnn.py. [Accessed: Mar. 24, 2020].

[19] Opencv, “opencv/opencv,” GitHub. [Online]. Available: https://github.com/opencv/opencv/blob/master/samples/dnn/tf_text_graph_mask_rcnn.py. [Accessed: Mar. 24, 2020].