Smart Application for Real Time Detection: An Improved Lightweight Detector for Real-Time Vehicle Detection

Ala Alsanabani
School of Artificial Intelligence
Xidian University
Xi'an, China

Ahed Abugabah
College of Technological Innovation
Zayed University
Abu Dhabi, UAE

Licheng Jiao
School of Artificial Intelligence
Xidian University
Xi'an, China

***Abstract*—**With the goal of creating a model that satisfies the demands of detecting vehicles on the road in real time and delivers high FPS speed and high accuracy, we propose enhancements to the YOLOv4-tiny detector in this work. We suggest a modification to the FPN to increase the semantic and location information between the higher and lower levels. Also, to further solve the issue with NMS, we employed soft-NMS in our model. Moreover, to address the drawbacks of the IoU, we proposed a way to improve anchor clustering. The experimental results show that our proposed model outperforms the basic model with 3.29% higher mAP and 1.17 higher FPS.

Keywords; Object detection, Tracking, Real-time object detection, Vehicle detection and tracking.

I. INTRODUCTION

Object detection and its sub-tasks, such as object classification and localization, multi-object detection, instance segmentation, object tracking, and scene understanding, have attracted a lot of attention for the purpose of solving life's daily difficulties [1], [2]. In recent years, a variety of useful solutions for addressing object detection issues have been established. The researchers' efforts have resulted in several changes to [3]'s initial structure in two primary paths. The first path is the improvement in model architecture. Over the years, researchers have developed several novel and innovative architecture designs that have significantly improved the accuracy of object detection systems. Some of the key improvements include Feature Pyramid Networks (FPN) [4], Multi-scale feature maps [5], Anchor-based detection [6], and Non-maximum suppression (NMS) [7]. The second path of progress is to deal with network dense connections in different ways to improve object detection accuracy at various scales [8], [9].

Real-time object detection is a computer vision task that involves detecting and locating objects in real-time with a low latency such that the output of the object detection system is produced in close to real-time. This is important for applications where immediate processing is required, such as surveillance systems, self-driving cars, and augmented reality. However, in order to meet the requirement of real-time processing, the models used in these systems are often optimized for speed, either by using lightweight models, running the models on specialized hardware (such as GPUs or TPUs), or using efficient algorithms for object detection [10]–[12].

Several methods and techniques have been developed to increase the accuracy of object detection systems, including two-stage object detectors like Faster R-CNN [6] and Mask R-CNN [13] and single-shot object detectors like YOLO [14] and SSD [15]. Many object detection systems have demonstrated impressive accuracy over a wide range of benchmarks. Nevertheless, this accuracy comes with a cost since these algorithms are computationally expensive and need a lot of resources to run and learn. This high computing cost can make it challenging to implement object detection algorithms in real-world applications. As a result, there is an increasing interest in designing lightweight object detection algorithms that are quick and efficient while still maintaining a high level of accuracy.

Moreover, regardless of the remarkable results in public datasets, real time vehicles detecting in real-world applications still poses a number of challenges due to several reasons such as; (i) occlusion where vehicles on roads can be partially or fully occluded by other objects, (ii) lighting variations where the lighting conditions on roads can vary greatly, especially during different times of day or in different weather conditions, (iii) high variability in vehicle appearance and (iv) distance variability where vehicles on roads can be at varying distances from the camera, making it difficult for a detection system to accurately detect and localize these objects [1], [11]. Current object detection algorithms tend to prioritize accuracy over speed, and as a result, many of the most accurate algorithms are computationally expensive and cannot run in real-time. To address this challenge, researchers are exploring ways to improve the speed of object detection algorithms while maintaining a high level of accuracy [15].

With the aforementioned obstacles in mind, the goal of this paper is to investigate and propose a real-time vehicle detection lightweight model. We suggest an enhanced YOLOv4-tiny [16] model based to detect vehicles in real time in light of compromising between accuracy and FPS. The contributions of our research are: (i) We propose an improvement on the FPN method to enhance the ability of the model to increase semantic and localization information between the higher and lower levels, (ii) We propose replacing the NMS with an alternative method to address the problem of removing the prediction box when there is a high overlap between two adjacent vehicles, (iii) We propose an improvement on K-means clustering to improve the model's ability to differentiate in aspect ratio and to improve its ability to determine how far apart two samples are from one another, and (iv) The D-VehU dataset is created to carry out the experiments we conducted for this paper.

II. Related Work

Authors of [17] proposed a low-cost method for detecting patterns in photos, and it discusses the performance concerns that [18] experienced. Rather than the utilization of key-point pairs, it adopts the usage of triplet key-points in order to localize objects, which enables the network to extract additional information out of the input image. In addition to that, a platform based on the benchmark of Mask Region-based CNN (R-CNN), is given, and it accomplishes state-of-the-art detection performance. [19] proposed using a weighted Bi-Directional Feature Pyramid network (BiFPN), a novel way to aggregate object features.

YOLO is a detector family that was first coined in 2015 [14]. YOLO is a cutting-edge algorithm that is known for being both highly efficient and incredibly compact, making it one of the greatest contemporary algorithms for real-time applications. YOLO converts a locating problem into a regression problem and detects the things in one stage. On the MS COCO dataset, YOLOv4 [16] achieved a running speed of around 65.7 frames per second and a precision of up to 43.5 % through using state-of-the-art approaches and tricks like Bag of Freebies (BoF) and several Bag of Specials (BoS) strategies. BoF increases the detector's precision with no increase in the implementation time. The BoS, on the other hand, increases object detection accuracy while only raising the cost of inference by a tiny amount. The authors have developed CSPDarknet-53 by incorporating Cross-Stage-Partial (CSP) connections into the backbone structure, as inspired by [19]. The basic idea of CSP is to utilize both the low-level and high-level features in a CNN to improve the performance of object detection. CSP uses a series of cross-stage partial connections (CSPs) that combine features from different stages of the network to improve the feature representation.

YOLOv4-Tiny: computing resources in real-world applications are limited to a small number of integrated GPUs or even embedded CPUs with restricted memory. Many academics have proposed lightweight object detection systems as a solution to this problem. The YOLOv4-tiny model is a smaller version of the standard YOLOv4. It consumes fewer computing resources and memory and finds things more quickly. While its detection accuracy is reduced, it is adequate for various applications like pedestrian detection, agriculture detection, and vehicle detection [20]. One of the essential differences that distinguishes the YOLOv4-tiny from its standard sibling is its smaller architecture network size. Instead of CSPDarknet-53 backbone, the YOLOv4-tiny is less deep and contains fewer layers, as shown in Fig. 1, for the purpose of reducing network complexity and computational costs. The head component has been decreased from 3 scale predictions (13x13, 26x26, and 52x52) as in the standard version to only 2 scale predictions (13x13 and 26x26).

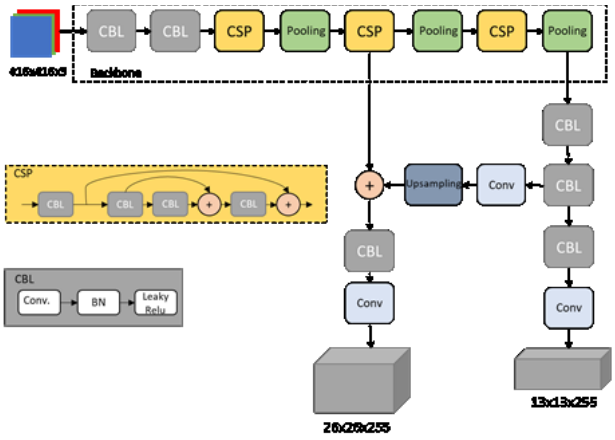

Fig. 1. YOLOv4-tiny network architecture.

Moreover, rather than utilizing the Path Aggregation Networks (PANet) and Spatial Pyramid Pooling (SPP) that the standard YOLOv4 model utilizes, YOLOv4-tiny uses a Feature Pyramid Network (FPN) to extract feature maps at different scales to enhance detection time. The network uses a bottom-up approach to extract features from the input image, and then uses a top-down approach to refine and combine these features into a single feature map. This allows the network to preserve both high-level semantic information and fine-grained details in the same feature representation, which is beneficial for detecting objects of different sizes.

III. Methodology

We came up with several techniques we did on the original model, inspired by some previous research [21], [22], that enhance our model's effectiveness and achieve better performance.

A. Improved FPN

The primary principle of the PAN approach, which is used by YOLOv4-tiny in the neck between the backbone and the detecting head, is to gather contextual information from different scales while boosting the acquisition of fine-grained features. In YOLOv4-tiny, F6 is made by placing a 1x1 convolution layer after C6. To facilitate concatenating with the C5 and retrieving the F5, the C6 is run and up-sampled. This procedure is depicted in Fig. 2. When this procedure is examined, it turns out that the PAN only uses one path to fuse the detailed information, and that path is from the lower level to the higher level.

To address the FPN's limitation of just having one path for information fusion, we propose adding a cross fuse to increase semantic and localization information between the higher and lower levels. The first phase of the proposed improved method uses the same technique as the PAN to extract F6 and F5, and then, before delivering the fused information to the detector's head, we add a cross fuse between the outputs of F6 and F5 to extract a stronger and richer representation of

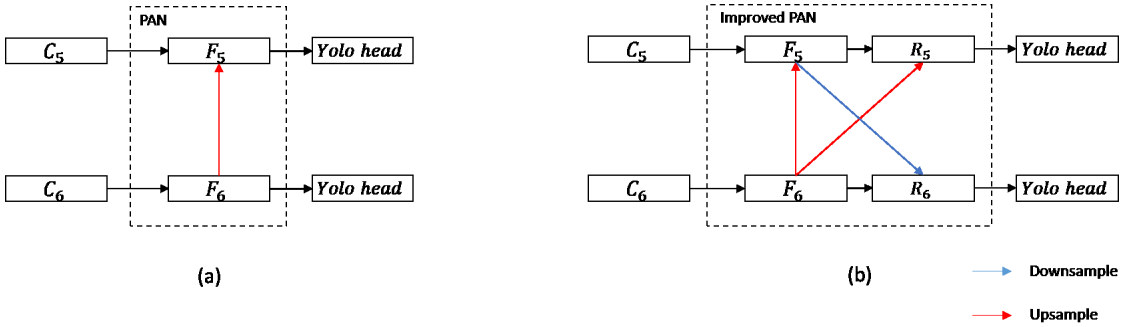

Fig. 2. The regular PAN used in the original work, (b) The improved PAN.

information in both R5 and R6. Where R5 is the result of the procedure of concatenating the F5 and the up-sampled F6, and R6 is the result of concatenating the F6 and down-sampled F5. Lastly, the values of R5 and R6 are used to feed the head of the detectors. Fig. 2 illustrates this process.

B. *Improved NMS*

The NMS mechanism depends on choosing the prediction box that has the highest confidence score and setting a threshold according to which it removes all nearby prediction boxes that overlap with the chosen predicted box by an amount higher than the threshold, while keeping the prediction boxes whose overlaps value are less than the threshold value. This process is represented by the mathematical equation (1):

$$s_i = \begin{cases} s_i, & IoU(M, b_i) < N_t \\ 0, & IoU(M, b_i) \geq N_t \end{cases} \quad (1)$$

Where b_i stands for the detected box, s_i stands for confidence score, M stands for the highest score value and N_t stands for the threshold. In applications for detecting vehicles on the roads in real life, it often happens that one vehicle is occluded by another vehicle from the viewpoint of the camera. This blocking sometimes results in the overlap between the two vehicle detection boxes being high. Here appears the problem of using the NMS that will remove the prediction box of the blocked vehicle. Fig. 3 shows the problem of removing the prediction box using NMS that if there is a high overlap between two adjacent vehicles, the NMS will delete the prediction box of the vehicle that is behind.

For this reason, and for the purpose of overcoming this problem, we adopted the use of Soft-NMS [23] in our model. The idea of the soft-NMS method is based on decaying the scores of the adjacent prediction box with a high score of confidence so that it avoids deletion when compared to the threshold value. The effect of this rescoring is higher for the neighboring prediction box with a high score of confidence than for those with a lower score of confidence. This rescoring is done by using two linear and Gaussian functions as shown in the equations (2) and (3) respectively.

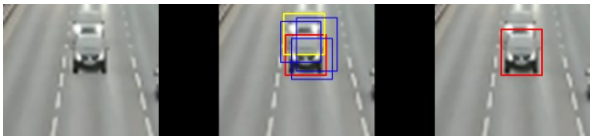

Fig. 3. Missed targets in NMS algorithm detection.

$$s_i = \begin{cases} s_i, & IoU(M, b_i) < N_t \\ s_i\left(1 - IoU(M, b_i)\right), & IoU(M, b_i) \geq N_t \end{cases} \quad (2)$$

$$s_i = s_i e^{\frac{-IoU(M,b_i)^2}{\sigma}}, \quad \forall b_i \notin D \quad (3)$$

In this way, whenever the overlap value is high and close to one, the penalty will be high, while it is lower, the lower the overlap value. Thus, we can avoid the drawback of the traditional NMS method and have a better ability to make the algorithm to deal with occluded vehicles.

C. *Improved k-means clustering*

The anchor box approach is used to increase the algorithm model's recall rate [6]. Basically, the idea of an anchor box is to specify a fixed-size set of initial boxes to be used to locate objects in the image. The precision and quickness of the model's detection will be significantly affected by the anchor box's design's since it effects on the loss function's ability to converge during the training of the model. The height and width of all actual boxes of the objects in the dataset has a key impact on the aspect ratio and the dimensions of anchor boxes. Consequently, for the sake of the a steady and expedite convergence during the training of the model, it is vital to choose the proper aspect ratio and anchor boxes sizes for various datasets.

The K-means clustering technique finds K groups or classes in the provided dataset by using distance as the similarity index, and sets a center for every class using the mean value calculated out of each group data points. Euclidean distance is used by the typical K-means to determine how far apart two samples are from one another. For clustering and choosing the possible box on the dataset (MS COCO dataset [24]) used by the original work, IoU was adopted by the original YOLOv4-Tiny algorithm.

In our work, we adopted the use of GIoU as it solves the dilemma that the traditional IoU suffers from which is represented in its inability to discern between distinct alignments of two objects. From Fig. 4 we can notice that although the aspect ratio is different for the green box in Fig. 4-a and 4-b, the IoU gives the same value because it is purely dependent on the result of dividing the intersection of the two boxes, red and green, by their union. Therefore, IoU is unable to differentiate between each aspect ratio, which results in difficulty for the network to determine which of the two boxes is more appropriate to the cluster box. On the other hand, the GIoU, due

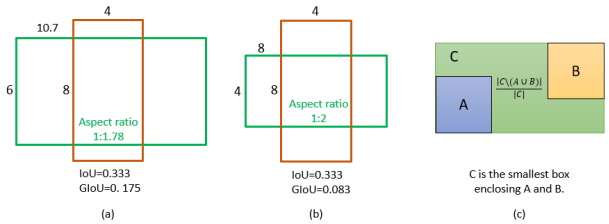

Fig. 4. A comparison showing the difference in the aspect ratio value. The IoU produces equal value for two different pixels as in (a, b) while the GIoU is able to differentiate between the two pixels and shows two different values. In part (c), the smallest box that includes both boxes show the mechanism for calculating the GIoU.

to its more comprehensive calculation method, gives different results when overlapping two different aspect ratio boxes with the ground truth box, as shown in the GIoU values in Fig. 4-a and 4-b. These different GIoU values make it possible to distinguish the aspect ratio and thus a better ability to know which boxes are the most appropriate. The green rectangular in Fig. 4-c is the smallest box that includes both boxes A and B, and it is the region that GIoU takes into account according to its calculation formula as in equation (4).

$$GIoU = \frac{|A \cap B|}{|A \cup B|} - \frac{|C \backslash (A \cup B)|}{|C|} \quad (4)$$

Where A stands for the sample box size, B for the cluster box size, C for the smallest size of the rectangle box that jointly encloses the sample box and the cluster box. And finally, the $C\backslash(A \cup B)$ donates the value that represents the area difference between the union of the two boxes A and B with the area of the box C.

IV. Experiments

A. Dataset

Taking into account the previously mentioned settings for installing the camera and calculating the interest region's distances, a dataset of eight video clips, two minutes long each, was created during daytime traffic activity on one-way public streets from four locations across the east of London, UK. Video clips were at 30 fps and at 1920 x 1080 resolution. For the first two minutes of each video clip, we captured screenshots at a rate of one per two seconds. By doing this, we were able to obtain 480 images. After removing any images that did not include vehicles, we were left with 453 images for our dataset, which we named D-VehU. We utilized the free tool LabelImg [25] to label the images before implementing some augmentation techniques. We made some augmentation at the pixel level, such as blurring and the difference in lighting, which reflects the realism of the detection on the ground. We also added some augmentation on the spatial level, such as the flip on the horizontal level. Thus, after this process, we ended up enlarging our dataset from 453 to 1359 images. The sizes of the images were modified to 416x416 pixels. To prevent corrupting the aspect ratio of the vehicles, black padding was added to the photos to retain their aspect ratios.

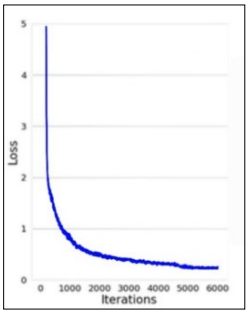

Fig. 5. Loss curve over 6000 iterations.

B. Results

In our experiments, in which we tested the model's performance on unseen images, we used the following hardware specification and testing settings; NVIDIA GeForce GTX 1060, 16 GB RAM, Core-i7 and the settings were batch size is 16, subdivisions is 16, learning rate is 0.00261, momentum is 0.9, decay is 0.0005, and the iterations was set to 6000. Looking at the loss curve during training, Fig. 5, it can be noted that the model achieved minimum losses smoothly and quickly during the training iterations, and the reason for this is due to the design of the model that uses the Leaky ReLU function and its high ability to learn. This is also a positive indicator that the model learned the characteristics of the vehicles well during the training period.

Both models produce above the minimum frame rate for real-time applications and meet real-time operating requirements (24 fps). By delivering a higher FPS of 54.06 as compared to the 53.43 achieved by the original model, the proposed model demonstrated superiority over the original model with an increase of 1.17%, demonstrating the usefulness and efficiency of the suggested adjustments in our model. Furthermore, the suggested model outperformed the original model in terms of mAP, with values of 96.66 and 93.53, respectively which represents an improvement of 3.29%. Fig. 6 shows a comparison of the results of the base model and the proposed model in terms of FPS and mAP.

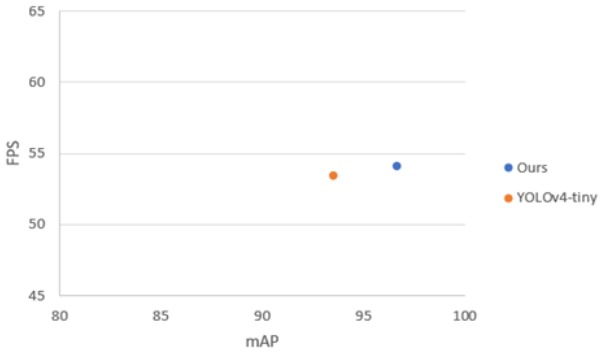

Fig. 6. A comparison of results in terms of FPS and mAP between the base model and the proposed model.

TABLE I. ABLATION STUDY RESULTS.

Improvements		Baseline	Exp.1	Exp.2	Exp. 3	Exp. 4
Improved FPN			√			√
Soft NMS				√		√
Improved clustering					√	√
Metrics	mAP	93.53	95.46	94.2	94.06	96.66
	FPS	53.43	53.36	53.33	54.23	54.06

We also examined the effect of the proposed improvements and their reflection on the performance of the model separately in the ablation analysis, and the results were as shown in Table I.

The results show that the improved clustering anchor played the major role to improve the FPS performance because the model's ability to distinguish the aspect ratio and thus a better ability to know which boxes are the most appropriate. Furthermore, the improved FPN and the adoption of soft-NMS played a better role in improving the mAP as a result of the model's higher ability to generate richer feature maps as well as better handling of adjacent predicted boxes of vehicles. Fig. 7 displays some of the results of the images that show the performance of the proposed model in detecting vehicles on the roads.

V. CONCLUSION

In this paper, we suggest an improvement to the YOLOv4-tiny detector with the aim of obtaining a model that meets the requirements of detecting vehicles on the road in real time. We proposed an improvement in FPN to increase semantic and localization information between the higher and lower levels. Moreover, we adopted soft-NMS in our model to address the problem of NMS which tends to remove the prediction box when there is a high overlap between two adjacent vehicles. We also suggested an improvement on anchor clustering by using the GIoU to overcome the shortcomings of the IoU. We trained and evaluated the suggested model on the D-VehU dataset. Our proposed model has 3.29% greater mAP and 1.17% higher FPS than the base model. The results showed that the proposed model could be a good fit for road vehicle detection applications.

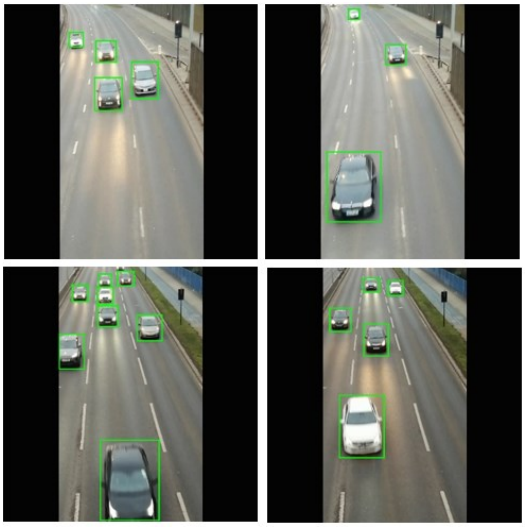

Fig. 7. Examples of detected vehicles.

REFERENCES

[1] L. Jiao et al., "A Survey of Deep Learning-based Object Detection," arXiv:1907.09408 [cs], Oct. 2019, doi: 10.1109/ACCESS.2019.2939201.

[2] J. Han, D. Zhang, G. Cheng, N. Liu, and D. Xu, "Advanced Deep-Learning Techniques for Salient and Category-Specific Object Detection: A Survey," IEEE Signal Processing Magazine, vol. 35, no. 1, pp. 84–100, Jan. 2018, doi: 10.1109/MSP.2017.2749125.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in Advances in Neural Information Processing Systems 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. Accessed: Dec. 26, 2019. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[4] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, Jul. 2017, pp. 936–944. doi: 10.1109/CVPR.2017.106.

[5] F. Yu and V. Koltun, "Multi-Scale Context Aggregation by Dilated Convolutions," arXiv:1511.07122 [cs], Nov. 2015, Accessed: Jun. 21, 2019. [Online]. Available: http://arxiv.org/abs/1511.07122

[6] 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, Jun. 2017, doi: 10.1109/TPAMI.2016.2577031.

[7] J. Hosang, R. Benenson, and B. Schiele, "Learning non-maximum suppression." arXiv, May 09, 2017. doi: 10.48550/arXiv.1705.02950.

[8] A. A. Alsanabani, S. A. Saeed, M. Al-Mkhlafi, and M. Albishari, "A Low Cost and Real Time Vehicle Detection Using Enhanced YOLOv4-Tiny," in 2021 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA), Jun. 2021, pp. 372–377. doi: 10.1109/ICAICA52286.2021.9498188.

[9] M. Mathieu, Y. LeCun, R. Fergus, D. Eigen, P. Sermanet, and X. Zhang, "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks," Dec. 2013, Accessed: Mar. 27, 2020. [Online]. Available: https://openreview.net/forum?id=Hq5MgBFOP62-X

[10] I. Baek, A. Davies, G. Yan, and R. R. Rajkumar, "Real-time Detection, Tracking, and Classification of Moving and Stationary Objects using Multiple Fisheye Images," in 2018 IEEE Intelligent Vehicles Symposium (IV), Jun. 2018, pp. 447–452. doi: 10.1109/IVS.2018.8500455.

[11] A. A. Alsanabani, M. A. Ahmed, and A. M. Al Smadi, "Vehicle Counting Using Detecting-Tracking Combinations: A Comparative Analysis," in 2020 The 4th International Conference on Video and Image Processing,

New York, NY, USA, Dec. 2020, pp. 48–54. doi: 10.1145/3447450.3447458.

[12] O. Köpüklü, X. Wei, and G. Rigoll, "You Only Watch Once: A Unified CNN Architecture for Real-Time Spatiotemporal Action Localization," Nov. 2019, Accessed: Dec. 22, 2019. [Online]. Available: https://arxiv.org/abs/1911.06644v1

[13] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in 2017 IEEE International Conference on Computer Vision (ICCV), Oct. 2017, pp. 2980–2988. doi: 10.1109/ICCV.2017.322.

[14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2016, pp. 779–788. doi: 10.1109/CVPR.2016.91.

[15] W. Liu et al., "SSD: Single Shot MultiBox Detector," in Computer Vision – ECCV 2016, Cham, 2016, pp. 21–37.

[16] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," arXiv:2004.10934 [cs, eess], Apr. 2020, Accessed: Nov. 08, 2020. [Online]. Available: http://arxiv.org/abs/2004.10934

[17] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "CenterNet: Keypoint Triplets for Object Detection," in 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Oct. 2019, pp. 6568–6577. doi: 10.1109/ICCV.2019.00667.

[18] H. Law and J. Deng, "CornerNet: Detecting Objects as Paired Keypoints," in Computer Vision – ECCV 2018, Cham, 2018, pp. 765–781. doi: 10.1007/978-3-030-01264-9_45.

[19] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and Efficient Object Detection," in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2020, pp. 10778–10787. doi: 10.1109/CVPR42600.2020.01079.

[20] X. Wang, S. Wang, J. Cao, and Y. Wang, "Data-Driven Based Tiny-YOLOv3 Method for Front Vehicle Detection Inducing SPP-Net," IEEE Access, vol. 8, pp. 110227–110236, 2020, doi: 10.1109/ACCESS.2020.3001279.

[21] Z. Jiang, L. Zhao, S. Li, and Y. Jia, "Real-time object detection method based on improved YOLOv4-tiny," arXiv:2011.04244 [cs], Dec. 2020, Accessed: Apr. 08, 2021. [Online]. Available: http://arxiv.org/abs/2011.04244

[22] J. Li, Y. Xu, K. Nie, B. Cao, S. Zuo, and J. Zhu, "PEDNet: A Lightweight Detection Network of Power Equipment in Infrared Image Based on YOLOv4-Tiny," IEEE Transactions on Instrumentation and Measurement, vol. 72, pp. 1–12, 2023, doi: 10.1109/TIM.2023.3235416.

[23] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis, "Soft-NMS -- Improving Object Detection With One Line of Code." arXiv, Aug. 08, 2017. doi: 10.48550/arXiv.1704.04503.

[24] T.-Y. Lin et al., "Microsoft COCO: Common Objects in Context," in Computer Vision – ECCV 2014, Cham, 2014, pp. 740–755.

[25] T. Lin, "labelImg: LabelImg is a graphical image annotation tool and label object bounding boxes in images." Mar. 30, 2019. Accessed: Mar. 30, 2019. [Online]. Available: https://github.com/tzutalin/labelImg