Multi-Class Multi-Movement Vehicle Counting Based on CenterTrack

Viktor Kocur Milan Ftáčnik
Faculty of Mathematics, Physics and Informatics
Comenius University in Bratislava
{viktor.kocur,milan.ftacnik}@fmph.uniba.sk

Abstract

In this paper we present our approach to the Track 1 of the 2021 AI City Challenge. The goal of the challenge track is to to analyse footage captured with traffic cameras by counting the number of vehicles performing various predefined motions of interest. Our approach is based on the CenterTrack object detection and tracking neural network used in conjunction with a simple IoU-based tracking algorithm. In the public evaluation server our system achieved the S1 score of 0.8449 placing it at the 8th place on the public leaderboard.

1. Introduction

Accurate and efficient analysis of traffic flows is an important component of intelligent transportation system design. Recent advances in camera technology and computer vision made it possible to efficiently and accurately analyse traffic flows based on video footage captured by traffic cameras. This task is the subject of the Track 1 of the 2021 AI City Challenge. Specifically the goal of the challenge is to design an algorithm which receives footage from a traffic camera observing an intersection and outputs the counts of two main types of vehicles for various possible movements of interest through the observed intersection in real-time on an edge device embedded with the camera. In this paper we describe our approach [1] to the challenge track which is based on the CenterTrack [26] object detection and tracking network used in conjunction with a simple IoU-based tracking algorithm.

Since CenterTrack is based on a single stage keypoint-based object detection network CenterNet [27] it is well-suited for applications with limited computational resources. Compared to the pure object detection network CenterTrack has an additional output for each detected object which denotes the displacement of the object from its previous position. We use this information to track the objects across multiple frames using a greedy algorithm based on the IoU metric. Since there are no annotations of vehicle bounding boxes or tracks provided for this challenge we used a model pre-trained on the MS COCO dataset [13].

To classify the tracks into various movements we use the 2D field modeling of routes as proposed by Yu et al. [25] in their submission to the 2020 version of the challenge. However we use simpler classification criteria. To classify the category of the vehicle we use non-maximum suppression for overlapping bounding boxes of cars, trucks and buses inspired by [25] as well. To determine the time when a tracked vehicle leaves the region of interest we use a simple linear regression on the positions of the suitable corner of the bounding box in the 2D completeness field for the movement.

Overall we present a conceptually simple approach to the Track 1 of the challenge. Our method achieved $S1_{Effectivness}$ score of 0.8882 and $S1_{Efficiency}$ score of 0.7439 resulting in the combined $S1$ score of 0.8449 placing our submission at the 8th place on the leaderboard.

2. Related work

Since the 2020 AI City Challenge [16] also included an almost identical challenge track in this section we mostly provide an overview of the various methods submitted for the 2020 challenge. We concentrate on the most important aspects of the various strategies used in the last year's challenge while mostly omitting minor tricks and smoothing mechanisms.

2.1. Vehicle Detection

In recent years the task of object detection has been dominated by methods based on convolutional neural networks. Most approaches use a general purpose backbone architecture to extract features from the images with an additional structure enabling the network to output bounding boxes and classes for the detected objects. In general these approaches can be divided into two major categories: single-stage and two-stage object detectors.

[1] The code is available online: https://github.com/kocurvik/aicc

Two-stage detectors first use a network to generate proposals for possible positions of objects. In the second stage the proposals are checked to determine if the proposals actually correspond to an object. At this stage the bounding boxes are also refined to better fit the objects. The most prominent example of a two-stage object detector is Faster R-CNN [20]. Its extension Mask R-CNN [9] also enables detection of masks of objects along with their bounding boxes. The two best scoring teams [14, 17] in the last year's challenge both used Faster R-CNN. Mask R-CNN was also used by two teams [6, 25].

As the name suggests single-stage object detectors output refined bounding boxes in one pass of the network. This can either be achieved with a structure of anchorboxes in methods such as YOLO [19] or RetinaNet [12]. YOLO was used by multiple teams [5, 7, 22] and an improved version of RetinaNet was used by one team [2]. A more recent type of single-stage object detectors treats object detection as a task of detecting keypoints of bounding box corners [11] or centers [27] with additional regression outputs used to fully define the bounding boxes. The latter method called CenterNet [27] was used by one team [21] in the challenge. In our approach we opt to use an extension of CenterNet called CenterTrack [26]. We opt to use this method based on its good speed-accuracy tradeoff.

2.2. Tracking

Kalman filter [10] has been a widely used method for tracking for several decades. A popular tracking algorithm SORT [3] uses a simple combination of the Kalman filter with the Hungarian algorithm to perform online tracking. Its extension DeepSORT [23] adds deep learned appearance based features to aid the tracking. DeepSORT has been used as a basis for multiple submissions to the last year challenge [2, 5, 7, 14, 22].

Alternatively some approaches [6, 17, 25] have used features or proposals extracted during object detection to aid in formation of tracks. These approaches either incorporate appearance information which requires no additional computational overhead since it is already computed by the object detection network [17, 25] or they use a standalone network to compute it [5].

Bochinksi et al. [4] have shown that a simple tracker based on the IoU metric of object bounding boxes in consecutive frames can outperform more complicated trackers when the objects are detected reliably. A similar IoU based strategy was employed by one of the teams in the last year's challenge [21].

In our approach we opt to use the CenterTrack network [26] which is an extension of the CenterNet [27] object detector. In addition to the current frame CenterTrack requires the previous frame as an additional input and outputs object bounding boxes along with the estimated displacement vector for bounding box center in the previous frame which can be used to improve tracking. Even though the principle behind the tracking mechanism is simple CenterTrack achieves state of the art perfomance on multi-object tracking benchmarks [8, 15]. In our method we use the displacement vectors in a greedy algorithm tracking scheme based on the IoU metric.

2.3. Vehicle type classification

As presented in subsection 2.1 all of the teams used deep learning based object detectors. Some teams [17, 25] used object detectors trained on the general purpose MS COCO dataset [13]. The definition of truck in the MS COCO dataset is different from the definition used by the challenge. This discrepancy could be resolved by considering the sizes of detected vehicles [17] or merging all of the detections into single class via NMS and deciding on the final class based on the proportions of MS COCO classes assigned to the given track [25]. In our method we use a very similar strategy to the latter approach.

Some teams have avoided this issue completely by fine-tuning a model pre-trained on MS COCO on a manually annotated dataset [14, 22] or by performing the whole training on a custom dataset [6, 21]. We found that models trained on MS COCO are sufficiently accurate and therefore we chose to not pursue this strategy.

2.4. Movement counting

Given the detected tracks it is necessary to identify the movement of interest they belong to as well as determining the frame during which the vehicles leave region in question. To achieve movement of interest assignment many methods have used manually annotated regions corresponding to whole movements [6, 21], pairs of entry and exit regions [7], multiple regions [5] or tripwire-like lines [17].

Other approaches assign the movements of interest based on distance of the track from a typical path [14, 25]. We opt to use a similar strategy by utilizing the 2D proximity and completeness fields proposed in [25]. However we use a simpler assignment criterion.

Almost all of the teams also propose some form of filtering or joining of tracks. Additionally most teams also use various forms of simple modelling to determine the frame during which the vehicle should be counted.

2.5. Efficiency

The main goal of the challenge this year is to investigate the potential of vehicle counting systems which can be deployed on edge devices ideally directly integrated within the camera setup. In the last year's challenge the top-5 systems [2, 14, 17, 22, 25] from the leaderboard have been benchmarked on a single machine with 32GB RAM, i7-5930K CPU (12 Cores) and one NVIDIA TITAN Xp GPU (12GB

memory). Even on this relatively high-end machine the systems reached FPS rates in the range of 6.41 [25] to 34.82 [22]. This was somewhat in contradiction with the leaderboard efficiency results from the first stage of the challenge as the teams which used multiple high-end GPUs to benchmark their results were at a significant advantage as their total execution times were relatively low and the automated efficiency base calculation script [16] failed to account for such setups.

To investigate how the 5 systems perform on edge devices Anastasiu et al. [1] attempted to run them on an Nvidia Jetson NX edge device. Due to technical limitations only two of the methods [17, 22] have been benchmarked successfully. Of these two only [22] performed with at least somewhat meaningful efficiency at around 5 FPS while the stream was at 10 FPS for most of the videos. This indicates that there is still room for improvement before these methods can be applied on edge devices.

In our approach we attempt to optimize our system with the use of a multithreaded approach to the overall data flow of the system and utilization of the automatic mixed precision inference which automatically runs the inference of the network with halved precision where applicable.

3. Method

In this section we present the details of the method we used for our submission to the challenge. The purpose of the proposed method is to analyze traffic flow captured by a video camera observing a road or an intersection by counting the number of vehicles which crossed the observed area and classify them into various types of movements (e.g. coming from left and turning right on the intersection).

3.1. CenterTrack

To detect the vehicles we use the CenterTrack [26] object detection and tracking network with the DLA-34 backbone [24]. The network is trained on the MS COCO dataset [13] with input images with resolution of 512×512 px. In addition to the standard object detection output the network also outputs displacement vectors for each detected object. These displacement vectors correspond to the estimated relative position of the object in the previous frame (see Fig. 1).

In order for this to work the network takes at input the current frame of the video as well as the previous frame. In some variants a third input is added which contains the heatmap with all of the detected centers of objects from the previous frame. However the off-the-shelf model trained on the MS COCO dataset we use does not require such input. Note that the MS COCO dataset is composed of still images and does not contain any tracking annotations. Therefore the network is trained using simple augmentation techniques which emulate object movement [26].

During inference we keep all vehicles which are detected with a confidence score above 0.2.

3.2. Vehicle class NMS

The MS COCO dataset [13] contains three classes of vehicles: car, bus and truck. In the challenge the category truck refers only to long freight and utility trucks whereas in the MS COCO trucks also refer to smaller vehicles including vans, SUVs etc. Additionally the CenterTrack detector sometimes outputs two or more overlapping bounding boxes for two or even three of these classes. To remedy this we perform non-maximum suppression of the spurious detections in a fashion similar to the one employed in [25]. Whenever two bounding boxes corresponding to the classes bus and truck overlap with IoU greater than 0.7 we discard the bus detection. Any bus detections which were not discarded are then reclassified as trucks. In the next step we check for overlaps between trucks (which now includes buses) and cars and we again discard any truck detections if the IoU is greater than 0.7.

3.3. Tracking

In the original implementation CenterTrack uses a simple greedy algorithm which associates objects in consecutive frames based on distances of bounding box centers. The distance is calculated between the center of the bounding box in the previous frame and the center of the bounding box in the current frame shifted by the displacement vector. This approach adds only a very small computational overhead over the base CenterNet architecture and is surprisingly effective. However we found this approach to not be suitable especially in very crowded scenes with vehicles of various sizes present. To remedy this we use the IoU metric of the object bounding boxes instead of distance of the centers. Similarly to the approach based on the centers we use the displacement vector to shift the bounding box in the current frame. This process is visualized in Fig. 1.

If a detection does not have IoU greater than 0.1 with bounding boxes of any of the active tracks which have not had a new bounding box assigned to them for that frame so far we either discard it if has a confidence score lower than 0.4 or we create a new track if this threshold is met.

During tracking we do not care about the labels of the vehicles e.g. one track can contain bounding boxes labeled as both trucks and cars.

3.4. Movement counting

When a track is considered for counting we first check if it contains at least 15 detections. Any tracks which do not reach this threshold are discareded. Next we calculate the ratio of detecions with the truck labels and car labels. If there are more than 30 percent truck labels we consider the vehicle to be of the truck class.

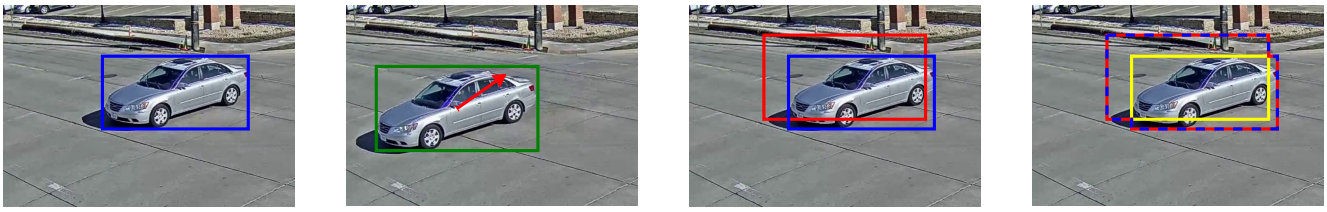

Figure 1: a) Detected bounding box (blue) of a vehicle. b) In the next frame the vehicle is detected with a bounding box (green) and a displacement vector (red). c) The displacement vector is used to shift the bounding box (green) detected in b to the estimated position in the previous frame (red). d) To determine whether to associate the two detections the IoU metric is calculated as the ratio of intersection (yellow) and union (blue-red) of the two bounding boxes.

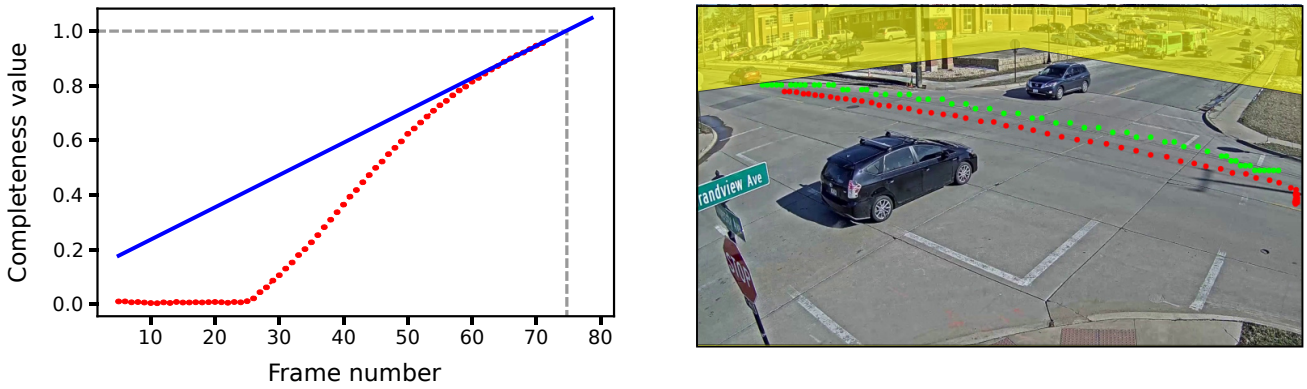

Figure 2: The red points represent the positions of the bottom-right corner of the bounding box of the vehicle which just moved outside of the region of interest. The blue line in the graph represents the regressed linear model of motion based on eight corner positions. The dashed gray line shows that the projected frame in which the vehicle should be counted is 74. In the image the green points represent the positions of the the center of the vehicle bounding box and the yellow overlay shows the part of image which lies outside of the region of interest for the camera.

To determine the movement of interest for the track we use the 2D proximity and completeness fields from [25]. However we design a different set of classification criteria. For a given image coordinate $\vec{x}$ the value of the proximity field P_i for the coordinate $P_i(\vec{x})$ represents the distance from the manually annotated path for a given movement i. Similarly the completeness field $C_i(\vec{x})$ represents what portion of the given path i would be passed e.g. the value is 0 at the start of the motion and 1 when the vehicle leaves the region of interest.

In a first step we eliminate the movements which are oriented in the opposite direction of the motion of the track thus keeping only a subset of valid movements:

$$V = \{i | C_1(\vec{x}_1) < C_n(\vec{x}_n)\}, \tag{1}$$

where $\vec{x}_1, \vec{x}_2 \ldots \vec{x}_n$ are the temporally ordered positions of the centers of bounding boxes for a given track. We then select the movement from V with the smallest cost D_i defined as

$$D_i = \overline{P}_i + 3 \cdot \sigma(P_i), \tag{2}$$

where $\overline{P}_i$ is the mean and $\sigma(P_i)$ is the standard deviation over the positions of the bounding box centers.

To determine the frame at which the vehicle leaves the region of interest we model the movement of the corner of the vehicle bounding box which should leave the region of interest last for the selected movement. Given the positions of the corners $\vec{y}_1, \vec{y}_2 \ldots \vec{y}_n$ and the corresponding frames $f_1, f_2 \ldots f_n$ we calculate

$$m = \underset{1 \leq j \leq n}{\arg\max}(C_i(\vec{y}_j)). \tag{3}$$

Afterwards we filter out results which do not conform to the following conditions:

$$C_i(\vec{y}_m) > 0.3, \tag{4}$$

$$C_i(\vec{y}_m) - \min_{1 \leq j \leq m} (C_i(\vec{y}_j)) > 0.25 \cdot \min\left(1, \frac{f_m}{50}\right). \tag{5}$$

The condition (4) prevents false positives for vehicles for which the tracking was interrupted in the beginning of the motion which is often the case when vehicle stops at an intersection and is then occluded by a vehicle passing through the intersection. The second condition (5) sets a threshold for minimal distance traveled to reduce the false positive rate. The minimum in the relation serves to slowly increase the threshold during the first 50 frames of the video. This is intended to prevent discarding of vehicles which are already close to exiting the region of interest at the start of the video.

Finally we apply linear regression to the values of $C(\vec{y}_j)$ and f_j for $j \in \{k|m-7 \leq k \leq m\}$. From the regression we obtain the parameters which represent the linear model of motion of the corner $P(\vec{y}_j) \approx a \cdot f_j + b$. We then calculate the frame at which the vehicle leaves the region of interest as the closest integer to the value $\frac{1-b}{a}$. An example of the values for a track and a resulting linear model is shown in Fig. 2.

3.5. Implementation details

The CenterTrack network is implemented in the PyTorch framework [18]. We run the network in the automatic mixed precision mode which should increase the inference speed especially on edge devices.

We run the system in a threaded fashion. The system spawns three threads one for loading frames and preprocessing the images, one for running model inference and one for processing the results of the inference and performing tracking. This approach significantly improves system efficiency.

4. Evaluation

We evaluated our method using the evaluation server hosted by the challenge organizers. The target metric is the S_1 score which is calculated as:

$$S_1 = 0.7 \cdot S1_{Effectiveness} + 0.3 \cdot S1_{Efficiency}. \quad (6)$$

The effectiveness score reflects the accuracy of the system and the efficiency score reflects its computational efficiency [16]. The final scores our system achieved are $S1_{Effectivness}$ score of 0.8882 and $S1_{Efficiency}$ score of 0.7439 resulting in the combined $S1$ score of 0.8601 placing our submission at the 8th place on the leaderboard.

5. Experiments

Since every team had only ten attempts at submission to the evaluation server there was only a limited room for ablation studies and parameter tuning. Sadly our first 4 submissions had an incorrect format and we thus could gain no relevant information from them. As can be seen in Table 1 our first correct submission achieved a relatively good result in terms of effectiveness we therefore directed our efforts at improvements in terms of efficiency. Replacing the serial execution of the system with a threaded one and a enabling the PyTorch automatic mixed precision led to a significant increase in efficiency while slightly decreasing effectiveness. The change in effectiveness score is reasonable since changing the inference precision of a neural network usually results in slight changes to the output. To achieve even greater speedup we ran the system for two video streams concurrently and fed the images into the network in a batched form. This led to further improvement in efficiency. The slight improvement of effectiveness was caused by fixing a bug in which the active tracks at the end of the stream were not processed. All of the experiments were run on a Google Cloud Compute Engine instance with an Nvidia T4 GPU and n1-highmem CPUs. Note that from the experiments which we did not submit to the evaluation server we found out that enabling automatic mixed precision has only a very slight positive effect on efficiency. We hope that the effect can potentially be greater when tested on an edge device, but we have no means of verifying this.

Type	Precision	$S1_{Effectiveness}$	$S1_{Efficiency}$	$S1$
S	FP32	0.8885	0.4661	0.8045
T	AMP	0.8734	0.6086	0.8016
TT	AMP	0.8882	0.7439	0.8601

Table 1: The results from the evaluation system for various implementations of our model. Type **S** refers to a simple serial pipeline, **T** refers to threaded pipeline and **TT** refers to threaded pipeline processing two videos at the same time. Precision FP32 refers 32-bit floating point precision during inference and AMP refers to the PyTorch automatic mixed precision.

6. Conclusion

In this paper we have presented the method developed for our submission to the Track 1 of the 2021 AI City Challenge. Our approach is conceptually simple while achieving reasonable effectiveness. Since the bulk of the computational complexity of our system lies in the CenterTrack network we see a potential for a further study of effectivness-efficiency tradeoffs when using different potentially more lightweight backbone architectures. Such study might uncover the best possible configurations in terms of effectiveness when faced with constrained computational power which might lead to efficient real-world applications.

Acknowledgments

This research was partially funded by project "Automatická kalibrácia dopravných kamier" provided by Comenius University in Bratislava UK/214/2021. The authors also gratefully acknowledge the support of NVIDIA Corporation with the donation of GPUs.

References

[1] David C Anastasiu, Jack Gaul, Maria Vazhaeparambil, Meha Gaba, and Prajval Sharma. Efficient city-wide multi-class multi-movement vehicle counting: A survey. *Journal of Big Data Analytics in Transportation*, pages 1–16, 2020.

[2] Bai B, Xu P, Xing T, and Wang Z. Dttm-vehicle-counting. https://github.com/Jilliansea/DTTM-Vehicle-Counting, 2020. GitHub repository.

[3] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)*, pages 3464–3468. IEEE, 2016.

[4] Erik Bochinski, Volker Eiselein, and Thomas Sikora. High-speed tracking-by-detection without using image information. In *International Workshop on Traffic and Street Surveillance for Safety and Security at IEEE AVSS 2017*, pages 1–6, Lecce, Italy, Aug. 2017.

[5] Nam Bui, Hongsuk Yi, and Jiho Cho. A vehicle counts by class framework using distinguished regions tracking at multiple intersections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 578–579, 2020.

[6] Ming-Ching Chang, Chen-Kuo Chiang, Chun-Ming Tsai, Yun-Kai Chang, Hsuan-Lun Chiang, Yu-An Wang, Shih-Ya Chang, Yun-Lun Li, Ming-Shuin Tsai, and Hung-Yu Tseng. Ai city challenge 2020-computer vision for smart transportation applications. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 620–621, 2020.

[7] Jan Folenta, Jakub Spanhel, Vojtech Bartl, and Adam Herout. Determining vehicle turn counts at multiple intersections by separated vehicle classes using cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 596–597, 2020.

[8] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012.

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2961–2969. IEEE, 2017.

[10] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[11] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision*, pages 734–750, 2018.

[12] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988. IEEE, 2017.

[13] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[14] Zhongji Liu, Wei Zhang, Xu Gao, Hao Meng, Xiao Tan, Xiaoxing Zhu, Zhan Xue, Xiaoqing Ye, Hongwu Zhang, Shilei Wen, et al. Robust movement-specific vehicle counting at crowded intersections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 614–615, 2020.

[15] Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. Mot16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*, 2016.

[16] Milind Naphade, Shuo Wang, David C. Anastasiu, Zheng Tang, Ming-Ching Chang, Xiaodong Yang, Liang Zheng, Anuj Sharma, Rama Chellappa, and Pranamesh Chakraborty. The 4th ai city challenge. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, page 2665–2674, June 2020.

[17] Andres Ospina and Felipe Torres. Countor: Count without bells and whistles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 600–601, 2020.

[18] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[19] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788. IEEE, 2016.

[20] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[21] Minh-Triet Tran, Tam V Nguyen, Trung-Hieu Hoang, Trung-Nghia Le, Khac-Tuan Nguyen, Dat-Thanh Dinh, Thanh-An Nguyen, Hai-Dang Nguyen, Xuan-Nhat Hoang, Trong-Tung Nguyen, et al. itask-intelligent traffic analysis software kit. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 612–613, 2020.

[22] Li Wenwei, Wang Haowen, She Yue, Dong Ke, Jiang Bo, Che Zhengping, Tang Jian, and Qiao Xiuquan. Aic 2020 challenge track 1 by orange-control.

https://github.com/liwenwei123/AIC_2020_Challenge_Track-1, 2020. GitHub repository.

[23] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE, 2017.

[24] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2403–2412, 2018.

[25] Lijun Yu, Qianyu Feng, Yijun Qian, Wenhe Liu, and Alexander G Hauptmann. Zero-virus: Zero-shot vehicle route understanding system for intelligent transportation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 594–595, 2020.

[26] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. In *European Conference on Computer Vision*, pages 474–490. Springer, 2020.

[27] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019.