Journal of Big Data Analytics in Transportation manuscript No.
(will be inserted by the editor)

Efficient City-Wide Multi-Class Multi-Movement Vehicle Counting: a Survey

David C. Anastasiu · Jack Gaul · Maria Vazhaeparambil · Meha Gaba · Prajval Sharma

Received: 15 Aug 2020 / Accepted: date

Abstract With the advent of accurate deep learning-based object detection methods, it is now possible to employ prevalent city-wide traffic and intersection cameras to derive actionable insights for improving traffic, road infrastructure, and transit. A crucial tool in signal timing planning is capturing accurate movement- and class-specific vehicle counts. To be useful in online intelligent transportation systems, methods designed for this task must not only be accurate in their counting, but should also be efficient. In this paper, we study the multi-class multi-movement vehicle counting problem, overview the state-of-the-art methods designed to solve this problem, and present a series of comprehensive experiments, using traffic footage with $O(5)$ vehicles captured from 20 different vantage points and covering various lighting and weather conditions. Our survey aims to answer the question whether we are ready to leverage traffic cameras for real-time automatic vehicle counting. The results of our analysis show several promising approaches and identify areas where additional improvement is needed.

David C. Anastasiu ✉
Computer Science and Engineering
Santa Clara University
E-mail: danastasiu@scu.edu

Jack Gaul
Mathematics and Computer Science
Santa Clara University
E-mail: jgaul@scu.edu

Maria Vazhaeparambil
The Harker School

Meha Gaba
Basis Independent Silicon Valley High School

Prajval Sharma
Cupertino High School

Keywords Traffic Analysis · Vehicle Detection and Classification · Vehicle Tracking · Vehicle Counting · Edge Computing

1 Introduction

Traffic analysis is an essential component of any intelligent transportation system (ITS) employed in a modern city. As the city grows and traffic patterns change, real-time adaptive traffic signal light control timing becomes a key tool to manage commute times and general travel times [34]. To be effective, these systems rely on accurate real-time counting of vehicles passing through intersections or traveling through highway corridors. The current approach for gathering these data include embedding induction loop detectors and other sensors in roadways, which cost millions of dollars and lead to weeks-long roadway closures when they need to be serviced.

With the advent of accurate deep learning-based object detection methods, a variety of methods have been devised for traffic analysis using already-existing city-wide traffic and intersection cameras to derive actionable insights for improving traffic, road infrastructure, and transit. These include object detection and classification [49,57,7,46], multi-camera vehicle tracking [30,36,28,29,70,15], traffic flow analysis [50,19,64], anomaly/accident detection [51,37,17,58,4], multi-camera vehicle detection and re-identification with real and synthetic training data [52,72,27,74,18,13,35,11], and city-wide multi-class multi-movement vehicle counting [52, 10,1,59,3,44,65,53]. Several challenges, such as the AI City Challenge [49,50,51,52] and the SUNY Albany UA-DETRAC benchmark [67], have been organized with the goal to accelerate the research and development of these techniques. The result has been hundreds of research teams from around the globe working to advance the state-of-the-art and bring these technologies to real-world implementation readiness.

In this paper, we focus our attention on the problem of city-wide multi-class multi-movement vehicle counting, with a goal to answer the question whether the technology is mature enough to be used in a real-world ITS. Solutions to this problem can help traffic engineers understand the traffic demand and freight ratio on select corridors, design better intersection signal timing plans, and improve traffic congestion mitigation. Moreover, real-time counting results could be used as input to adaptive online AI-based ITSs. In our analysis, we first survey the important work done in this space, paying close attention to five of the top methods that competed on this problem in the most recent AI City Challenge, and then we perform an in-depth evaluation of these methods to identify their strengths and weaknesses.

The paper is organized as follows. We first give a brief overview of the AI City Challenge and its work to advance the state-of-the-art in ITS problems (§ 2). Then, we survey the work done to advance video-based vehicle counting methods (§ 3). We describe the dataset and experimental evaluation methodology we followed in this study (§ 4) and then analyze the results of our experiments (§ 5), drawing conclusions from these analyses. A discussion on insights and future research (§ 6) and a short conclusion (§ 7) end the article.

2 The AI City Challenge

The AI City Challenge and associated workshops [52, 51, 50, 49] were designed by a group of academics from Iowa State University, San Jose State University, Santa Clara University, SUNY Albany, University of Washington, University of Maryland, Australian National University, among others, and collaborators from NVIDIA Corporation and Amazon, with the specific aim of improving the state-of-the-art of ITS problems. Over the past four years, the challenge has attracted hundreds of researchers each year who have worked to solve increasingly difficult problems.

2.1 A Short History

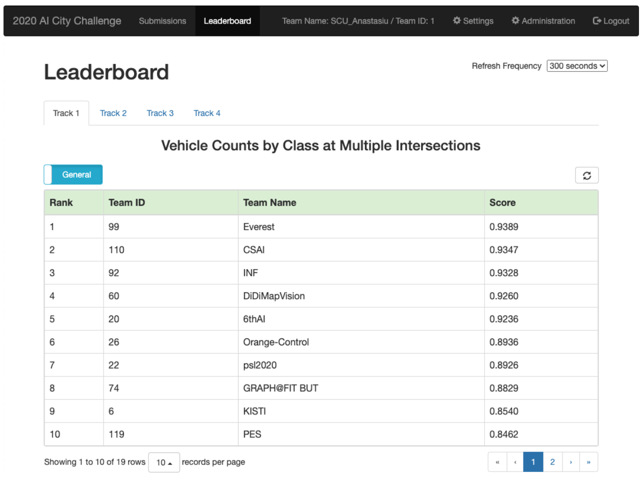
In 2017, the main problem in the space of deep learning-based traffic analysis was the lack of data needed to train accurate models. The challenge organizers thus constructed a collaborative image annotation platform and asked challenge participants to help build the dataset they would ultimately use in the challenge. Overall, 28 teams and more than 150 volunteers collaboratively annotated more than 150,000 key frames extracted from over 80 hours of traffic video captured at various intersections in four major U.S. cities. Each bounding box was assigned one of 14 different classes. In total, the teams contributed over 1.4M annotations which were then used by teams to train object detection and classification as well as vehicle tracking models. The challenge also emphasized the need for efficient edge computation capable models by encouraging teams to execute their models on Jetson TX2 development kits. Several teams developed methods that could ultimately run in real-time on the Jetson TX2, albeit at reduced frame rates (3–8 fps).

The 2nd and all subsequent editions of the AI City Challenge have been organized as CVPR workshops. Each continued to push the development of ITS methods and systems. In 2018, the organizers added challenge tracks for estimating traffic flow characteristics, such as speed, detecting anomalies caused by crashes or stalled vehicles in diverse weather conditions, and multi-camera tracking and object re-identification in urban environments. While collaborative annotations were no longer employed, the organizers created and annotated several new datasets for the challenge using innovative techniques such as GPS and time-synchronized scout vehicles embedded in traffic. New tracks and datasets were once again added in the 2019 edition of the AI City Challenge [51], including the *CityFlow* dataset [63] which tackled the extremely difficult problems of city-scale multi-camera vehicle re-identification and tracking. Despite a great effort by many teams, these problems continue, even a year later, to receive the lowest performance scores among all AI City Challenge tracks.

The 4th edition of the challenge [52] has once again pushed the development of ITS, first by introducing the multi-class multi-movement vehicle counting track, and second by evaluating, for the first time, not only the performance but also the efficiency of the proposed methods for this track. This was the first time that a challenge combined effectiveness and efficiency evaluation of tasks needed by the Department of Transportation (DOT) for operational deployments of ITS systems. We focus on the multi-class multi-movement vehicle counting problem for the remainder of this article.

2.2 Evaluation

One of the authors, David C. Anastasiu, has served, since the onset of the challenge, on the AI City Challenge organizing committee and additionally as the Eval-



Leaderboard

Refresh Frequency: 300 seconds

Vehicle Counts by Class at Multiple Intersections

Rank	Team ID	Team Name	Score
1	99	Everest	0.9389
2	110	CSAI	0.9347
3	92	INF	0.9328
4	60	DiDiMapVision	0.9260
5	20	6thAI	0.9236
6	26	Orange-Control	0.8936
7	22	psl2020	0.8926
8	74	GRAPH@FIT BUT	0.8829
9	6	KISTI	0.8540
10	119	PES	0.8462

Showing 1 to 10 of 19 rows 10 records per page

Fig. 1 The AI City Challenge **online evaluation system** leader board.

uation Chair for the event. In this capacity, as a way to encourage continuous improvement of the participating teams' methods, Anastasiu designed an **online evaluation system** that allowed multiple results be submitted by each team for each challenge track and automatically measured the effectiveness of results upon submission. In the vehicle counting track, teams were allowed a maximum of 5 submissions per day and 10 maximum total valid submissions.

In the multi-class multi-movement vehicle counting track, teams had to separately count two-axle vehicles and freight trucks following a set of predefined movements in videos from multiple stationary traffic cameras. Movements included left-turns, right-turns, and through traffic at one or more intersection approaches. To maximize the utility of the developed methods, teams were asked to develop not only effective but also efficient methods that could potentially work in real-time. These methods were thus evaluated both on their ability to assess the correct number of type- and movement-specific vehicles in each video and also on their overall execution time.

To further encourage participant competitiveness, the evaluation system for the AI City Challenge has been designed to show the top-3 best scores on the leader board for each track (without revealing the teams' identifying information). However, the results shown while the challenge was in progress were computed on a 50% subset of the test set for each track, as a way to discourage excessive minor model adjustments to improve performance. The scores computed on the full test set were revealed, along with the team names and the remaining results, after the challenge submission deadline. Fig. 1 shows the final leader board for the vehicle counting track after the challenge completion.

2.3 Participants

The AI City Challenge has grown year-over-year, from 28 teams in 2017 to 315 teams in 2020, composed of 811 individual researchers from 37 countries. Of these teams, 233 downloaded the vehicle counting dataset, and 31 teams executed 215 submissions to the challenge evaluation system. An analysis of the overall submissions to the challenge in 2020 showed that 71% of the teams improved their initially submitted scores by at least 10%, validating the utility of the online evaluation system in pushing teams to innovate. In the remainder of the article, we will discuss some of the results submitted to the challenge by these teams.

3 Methods

Vehicle counting is not a new problem. Its benefits to improving city transportation has attracted a lot of research in this area over the years [34]. The problem has been solved classically through sensors added in or on top of the road, such as inductive loop detectors, pneumatic road tube sensors, acoustic detectors, piezoelectric, magnetic, or passive infrared sensors, to name just a few [23]. These sensors are, however, either temporary or very expensive to install and/or maintain. The prevailing solution in recent years has been to utilize existing signals form traffic video cameras in order to detect and count passing vehicles.

3.1 Counting by Regression

An earlier version of the vehicle counting problem asked only what the volume of traffic on a road was, which was initially solved through a series of regression-based methods [12,38,43]. Many of this type of methods can be successfully applied to both traffic/vehicle density estimation as well as people/crowd density prediction.

The effectiveness of the methods is greatly affected by the quality of the input, including image size, cleanliness of the camera lens, and distance between the camera and the vehicles being counted. As part of a larger ITS, Ignatius Moses Setiadi et al. [33], built a vehicle counting system that relied on classical computer vision techniques such as background subtraction and edge detection to estimate vehicle counts from CCTV footage at multiple locations in the city of Semarang, Indonesia. They found that both the captured image quality as well as frame rate play a big role in accurate counting estimation, especially when vehicles travel at high speeds.

Mirthubashini and Santhi [45] surveyed various vehicle detection and tracking techniques, including both deep-learning and classical detection and tracking methods, such as Gaussian mixture models (GMM) [60] and Mixture of Gaussians (MoG) + Support Vector Machines (SVM) [2]. They found that deep learning techniques for object detection have an advantage compared to conventional image processing techniques.

The majority of recent video-based vehicle counting methods follow a similar two-step approach. They first employ a single-camera multiple-object tracker to identify individual vehicles moving through the scene, then count each one as they traverse a virtual barrier identifying the region of interest (ROI) exit point. As these methods have matured, the vehicle counting problem has also increased in its complexity. In addition to counting vehicles on the road, methods are now asked to both classify vehicles in two or more classes (generally distinguishing between large trucks or busses and regular vehicles) and also separately count vehicles of each class as they exit the ROI across multiple routes or movements of interest (MOIs) [52].

3.2 Counting by Tracking

Most vehicle and object tracking in video methods follow a detect-then-track approach which relies on object detectors to identify/localize and potentially classify the objects, followed by methods that associate the identified objects across frames. As early as 2017, leveraging a dataset of over 1.4 million annotations of more than 150,000 video key frames obtained through a collaborative annotation process during the first AI City Challenge [49], Bhandary et al. [7] showed that transfer learning is an effective technique for object detection and classification in traffic videos.

Object Detection. Many tracking algorithms continue to use detectors based on classical computer vision techniques, such as background extraction. Rosas-Arias et al. [56], for example, devised a method for detecting changes in consecutive frames of video sequences based on incremental subspace learning which, coupled with statistical analysis of sequential frames, can be used to count vehicles that traverse a virtual threshold in the video. While the experimental evaluation was based on a fairly limited number of vehicles, the authors found the method to be both effective and efficient, able to process video data at up to 26 fps on a commodity machine.

Chen and Hu [14] devised a counting method that is robust to illumination changes and shadows. When tested on 66.16 minutes of 30 fps videos across 4 highway and city roadway segments with 2-4 lanes of moving traffic, the proposed motion-based detection method, which relies on background and shadow scene subtraction, achieved an impressive 99.93% accuracy. However, the method was not tested in diverse traffic conditions (e.g., stop-and-go traffic), at intersections, across diverse movements, at different times of day or in different weather scenarios. Moreover, the authors did not specify the resources used to process the videos and whether real-time processing is possible using commodity hardware.

When trained with enough data, convolutional neural network (CNN)-based models have shown to outperform classic techniques in object detection. These include methods such as YOLOv3 [54], Fast-RCNN [22], Faster-RCNN [55], Mask-RCNN [25], CenterNet [73], SSD [42], EfficientDet [61], and NAS-FPN [20], which have proven resilient for both general object detection and specifically for vehicle localization and classification.

Multi-Object Tracking. A variety of tracking algorithms have been devised over the years. In an effort to optimize efficiency, some trackers [8,9,32,31] rely only on positional information derived from the detectors, using predefined overlap thresholds to decide whether the vehicle is one encountered in the previous frame. While they are very efficient, these trackers tend to loose vehicles when they become obscured by, for example, a light pole, and create new tracks for the vehicles when they are "re-aquired". Recent methods, such as MDNet [48], TCNN [47], DeepSORT [68], TC [64], MOANA [62], and Tractor [5], use sophisticated visual feature matching techniques, along with position estimators, to improve tracking and minimize vehicle ID changes. In general, the methods construct tracklets, which are short sequences of associated vehicle bounding boxes across several sequential frames, and then employ optimization strategies for associating related tracklets to form tracks.

Vehicle Counting. The final step in the vehicle counting task is classifying the vehicle and associating its track with one of the MOIs. Some methods rely on the detector (e.g., YOLO) to also classify the objects. Trackers whose detector cannot be trusted to classify the vehicles use statistical information gathered from multiple frames to decide whether a track belongs to a car or a truck.

For the movement assignment, many methods compute a measure of similarity between the track and the set of predefined MOIs, which are themselves encoded as trajectories. The similarity can be based on the MOI and vehicle entry and exit points within the ROI [10, 59], or may use more complex path-based matching techniques [65,53,11]. Instead of manually encoding the MOI trajectories, some methods cluster the tracks to automatically discover potential MOIs [1,71,3,44]. Fig. 2, for example, shows discovered MOIs in several videos by the ENGIE method [53].

The 4th AI City Challenge [52] introduced a novel dataset for multi-class multi-movement vehicle counting and challenged teams to not only develop effective vehicle counting methods but also ensure their methods are efficient. In the following paragraphs, we will summarize the top-5 performing methods for the vehicle counting track in the challenge.

Didi-OC. Didi-OC, one of two teams from Didi AI Labs, used the YOLOv3 [54] network for detection and classification of the vehicles. The strength of YOLOv3 comes from its short inference time which stems from that fact that it is a single stage method. For tracking, Didi-OC used SORT [6]. The SORT algorithm associates new detections with ongoing tracks based on the dimensions of the bounding box. This is done using a Kalman filter. SORT is very well suited for online tracking as it is fast and accurate. The moment a track passes one of the lines of the ROI, the vehicle is counted.

Didi-MV [3]. The second DiDi team implemented a Detection-Track-Count algorithm for this challenge. For their detection algorithm, they used NAS-FPN [21] which is based on RetinaNet [40]. Often, deep learning detection algorithms can produce errors in edge cases such as low light and bad weather. To remedy this, they implemented a background model based on Hybrid Gaussian to extract the moving vehicles. A version of DeepSort [69] along with post processing was used for the tracking portion of the pipeline. Tracklets are divided into segments and a trajectory similarity score is calculated based on the inverse of an Euclidean distance-based metric and cosine similarity between each segment and each trajectory. Once the movement is assigned, a counting algorithm is applied. For incomplete tracks, Didi-MV implemented an algorithm to piece together the disjointed tracks until all short trajectories matched a corresponding vehicle.

ENGIE [53]. The Engie team used a Detection-Tracking-Counting pipeline in this challenge. They used Faster-RCNN [55] for their detection algorithm and used weights pre-trained on the COCO dataset [41]. COCO's dataset does not perfectly match the labeling of the challenge so it requires some additional filtering to correctly label the vehicles. The specific issue lies in the fact that pickup trucks are labeled as trucks in the COCO data set while the AI City Challenge classifies pickup trucks as cars. The solution the ENGIE team implemented relies on the assumption that freight trucks, specifically their bounding boxes, are at least 3 times larger than the smaller cars. The mean area of a track's bounding boxes is found. The average area of 90 percent of the smallest vehicles is used as a standard and compared to all trucks. Any vehicle that is at least 3 times this value is classified as a freight truck. With regards to filtering detections in general, they utilized 4 metrics to ignore invalid detections: ROI percentage overlap, minimum score, non max suppression, minimum area, and maximum area.

For the counting portion of the algorithm, Engie used a line crossing algorithm outlined in Cormen et al. [16] which differentiates the orientation of intersection vectors. The first and last detection for each track are used to define a vector for the track. They defined a set of calibration vectors that should not be crossed during the life of a track. Similarly, they defined another set of vectors that should be crossed during the life of a track. Tracks are then assigned to movements based on which vectors they did and did not cross. With each track assigned to a movement and classified as either car or truck, the last detection's timestamp of each track is used to count the vehicle.

CMU [71]. The CMU team used Mask R-CNN [24] for detection and tracking. In their method, the features extracted from each frame are used in an association algorithm to get vehicle IDs [66]. A few techniques were used during the identification process to help improve the efficiency of the processing pipeline. Detections of vehicles that were parked or stopped at a light were removed if their velocity was below a threshold. Also, a region filter was applied to exclude detections from outside the ROI. The method uses a few metrics to assign each track to a route. First, a distance metric is computed between each track and each route based on a proximity field for each route. The distance is further scale-normalized to account for differences in vehicle sizes as the vehicle moves closer to or away from the camera. A completeness metric is also used to track progress along the possible routes. Lastly, a stability metric is used to measure the distance throughout the process of route identification. The distance between a track and its correct route should be relatively constant. The overall proximity metric is thus designed to tell when tracks are entering or leaving MOIs.

Baidu [44]. The Baidu team also implemented a Detection-Track-Count algorithm for the challenge. They used Faster-RCNN [55] as their detector and Resnet50 [26] with FPN [39] as their backbone feature extractor. A pretrained COCO model was used and fine-tuned with manually annotated data from the Track-1 videos. For the tracking section of their pipeline, Baidu used DeepSort [69] as their baseline tracker. They implemented a similarity matrix based on a color histogram feature, a motion feature, and a shape feature. They used the Hungarian algorithm to match detections and tracklets based on this matrix. A shape based approach was used to assign tracklets to movements. They find the Hausdorff distance between each tracklet and movement as well as the direction similarity between each tracklet and movement which is defined as the vector that points from the initial bounding box to the final one. Lastly, the method manually defines spacial constraints for each movement that a tracklet must follow in order to be assigned. The counting is done when a tracklet leaves the ROI.

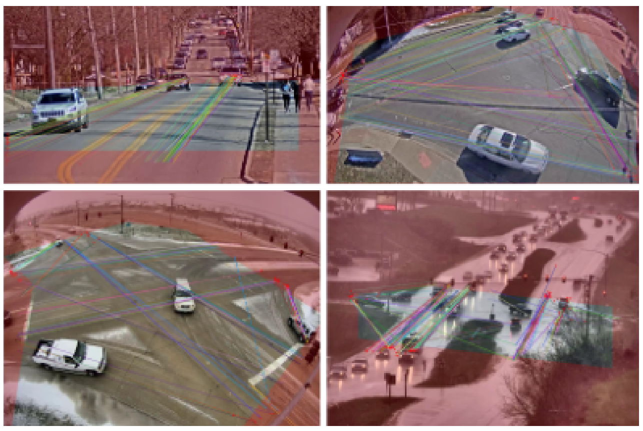


Fig. 2 Clustering of movements examples for the ENGIE method.

In the remainder of the article, we will thoroughly evaluate, under multiple scenarios, the effectiveness and efficiency of the top-5 submissions to the vehicle counting track of the 4th AI City Challenge.

4 Evaluation Methodology

The purpose of our experimental evaluation is to determine the state of readiness for real-world implementation of state-of-the-art multi-class multi-movement vehicle counting methods. The ideal implementation would leverage edge compute devices to obtain real-time per-movement vehicle counts for each class of vehicles, resulting in very small data payloads needing to be transmitted to a central server and/or to neighboring edge devices. In this section, we will describe our evaluation methodology, including the data, environments, metrics, and the particulars of the implementation used to validate the efficiency and effectiveness of each of the methods under comparison.

4.1 Dataset

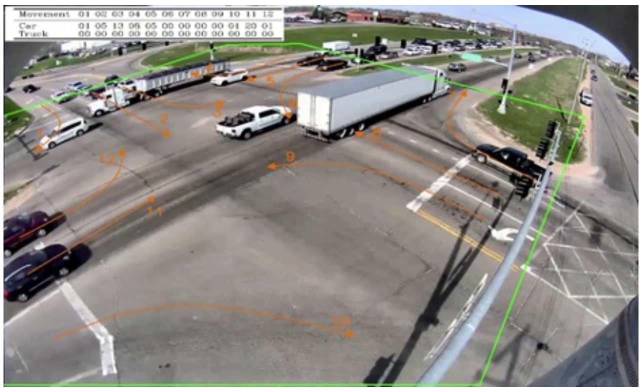


Fig. 3 An example scenario from the AI City Challenge **vehicle counting dataset** designed for multi-class, multi-movement vehicle counting.

We evaluate each method using Dataset A from Track 1 of the 4th AI City Challenge, which consists of roughly 5 hours of video captured in 20 unique scenarios and a variety of illumination and weather conditions. The footage is split into a total of 31 videos, of which 6 videos (29.35 minutes) have *rain*, 4 videos (20 minutes) were filmed at *dawn*, 1 video (5 minutes) has *snow*, and 20 videos (243.64 minutes) have *day*light conditions. Videos were captured at 960p or better resolution and most have 10 fps (except 2 at 8 fps and 2 at 15 fps).

For each video, teams were provided with visual and textual representations of ROIs and MOIs that are relevant to the vehicle counting task. The variety of scenarios, weather conditions, and video encoding provides a challenging realistic task for teams what wish to do well in solving this problem. Fig. 3 provides an example scene from one of the vehicle counting videos, with the ROI marked with a green polygon and the MOIs marked using orange arrows. Finally, vehicles were split into two classes, where medium and large tucks (e.g., moving trucks, garbage trucks, tractor trailers) were labeled “truck” and all other two-axle vehicles were labeled “car”.

The methods that performed well in the online evaluation system on Dataset A were invited to submit codes and running instructions for further evaluation on a held-out Dataset B, which consisted of 31 similar videos as those in Dataset A but totaling only roughly 4.5 hours. These evaluations were performed on a commodity system with an Intel i7 CPU and a single Nvidia Titan Xp GPU. The final challenge winners for the vehicle counting track were chosen as the top-2 performing methods on Dataset B.

4.2 Environments

To thoroughly understand the execution characteristics of the methods under comparison, we evaluated each method on four different servers and three edge devices, each with different CPUs and GPUs. In this section, we describe these systems and their characteristics, which were chosen to showcase the current limits of the state-of-the-art vehicle counting methods. To simplify the presentation and to easily differentiate between the systems, we name each system by the type of GPU installed in the systems. Table 1 provides a summary of the systems characteristics. Servers are listed at the top of the table, followed by edge devices. The *GPUs* column shows the number of GPUs in each server and the GPU architecture for each Jetson device. The *CUDA cores* column shows the number of CUDA cores per GPU in the given system. The *GPU RAM* column shows the amount and type of random access memory available to the GPU. The *Threads* column shows the number of physical CPU cores in the system. The *RAM* column shows the amount of random access memory available to the CPU. The Jetson systems share the same RAM space with the on-board GPU. Finally, the *Drive* column shows the type of storage device the operating system and experiment data reside on in the system.

In order to minimize I/O delay in the experiments, each system was outfitted with efficient drives capable of at least 500 MB per second reads, with the exception of the Jetson Nano edge device which did not have this capability and used an SD Card as its primary OS and data drive capable of up to 130 MB per second reads. Other than the Tesla V100 system, which used CentOS 7.4 as its operating system (OS), all systems relied on Ubuntu 18.04.4 LTS. All systems had CUDA 10.2 installed, although some methods relied on earlier versions of CUDA, which were provided via virtual environments. All Jetson development kits used the JetPack 4.4 software development kit, which included the latest Linux Driver Package (L4T) for the ARM processor, and CUDA-X accelerated libraries for CUDA 10.2 with built-in APIs for deep learning and computer vision. In addition to the CUDA cores listed in the 3rd column of Table 1, the Tesla V100, Titan RTX, and Jetson NX have 640, 576, and 48 tensor cores, respectively, which can further accelerate deep-learning computations.

4.3 Metrics

In this study, we are interested in both the effectiveness and efficiency of vehicle counting methods. We follow the technique in [52] for measuring the effectiveness of each method. The *effectiveness* score $S1_{\text{effectiveness}}$ is computed as a weighted average of the movement- and class-specific effectiveness for each video. Each video is split into k segments and a weighted root mean square error (wRMSE) between the predicted and true cumulative vehicle counts in each segment is computed as the local effectiveness. We used $k = 10$ in our experiments. To further reduce the impact of labeling errors that may have occurred in early segments, the wRMSE score weighs each record incrementally in order to increase the importance to later recorded cumulative counts.

$$\text{wRMSE} = \sqrt{\sum_{i=1}^{k} w_i(\hat{x}_i - x_i)^2}, \quad \text{where } w_i = \frac{i}{\sum_{j=1}^{k} j} = \frac{2i}{k(k+1)}. \tag{1}$$

The value x_i is the true cumulative vehicle count in all segments up to the ith segment, inclusive, and $\hat{x}_i$ is the prediction of that value. The wRMSE score is finally normalized based on the true vehicle count n of the given class in the given movement and video, as

$$\text{nwRMSE} = \max\left(0, \frac{\text{wRMSE}}{n}\right). \tag{2}$$

The final effectiveness score is the weighted average across all nwRMSE scores, with each weight being the fraction of all the true vehicles across the entire test set, i.e.,

$$S1_{\text{effectiveness}} = \sum_{v,m,c} \frac{n_{vmc} \times \text{nwRMSE}_{vmc}}{\sum_{v,m,c} n_{vmc}}, \tag{3}$$

for all videos v, movements m, and vehicles classes c in the set. Note that the score can be computed for the entire test set or for a subset of it, such as a single video or videos with certain characteristics (e.g., with rainy weather).

We measure *efficiency* as the wall clock running time of the method's execution (including reading videos and writing results) on 7 different evaluation systems. Execution time and resource utilization was measured using the GNU *time* routine. Since all methods are executed on the same systems, the execution times on each

Table 1 Characteristics of Evaluation Systems

System/GPU	GPUs	CUDA cores	GPU RAM	CPU	Threads	RAM	Drive
Tesla V100	2	5,120	32 GB HBM2	2x Xeon Gold 6148	80	768 GB	SSD RAID
Titan RTX	4	4,608	24 GB GDDR6	i9-7920X	24	128 GB	NVMe
Titan Xp	2	3,840	12 GB GDDR5X	i9-9900KF	16	64 GB	NVMe
Titan X	2	3,072	12 GB GDDR5	i9-9900KF	16	64 GB	NVMe
Jetson NX	Volta	384	8 GB LPDDR4x	Carmel ARM v8.2	4	–	NVMe
Jetson TX2	Pascal	256	8 GB LPDDR4	ARM A57	4	–	SSD
Jetson Nano	Maxwell	128	4 GB LPDDR4	ARM A57	6	–	SD Card

system can be compared to gain insight with regard to their relative efficiency. The $S1_{\text{efficiency}}$ score is based on the total execution time of each method normalized within the range [0, 5x video play-back time].

$$S1_{\text{efficiency}} = 1 - \frac{\text{time}}{5 \times \text{video total time}}. \tag{4}$$

We depart from the AI City Challenge evaluation of these methods in that we evaluate each video in the dataset independently. Some codes were engineered to use multiprocessing and potentially multiple GPUs to process several videos at the same time, which provides them an advantage against codes that did not rely on this engineering technique. Moreover, since we are interested in the ability of these methods to be executed on the edge, on systems with limited available processing and accelerator resources, we believe our evaluation of each method for one video at a time provides a fair comparison of the model's effectiveness and efficiency. This strategy also provides the added benefit of being able to analyze each methods' performance across different types of lighting and weather conditions, which was not possible using the challenge evaluation.

Finally, the overall evaluation score ($S1$) is a weighted combination between the efficiency score ($S1_{\text{efficiency}}$) and the effectiveness score ($S1_{\text{effectiveness}}$).

$$\begin{gathered} S1 = \alpha S1_{\text{efficiency}} + \beta S1_{\text{effectiveness}}, \\ \text{where}\, \alpha = 0.7, \beta = 0.3. \end{gathered} \tag{5}$$

4.4 Implementation

In this work, we have chosen to evaluate the top-5 methods with the best overall performance on the held-out Dataset B in Track 1 of the 4th AI City Challenge. In this section, we describe the implementations of the five methods under comparison. As noted previously, each method was modified to allow execution over a single video instead of processing all 31 videos at the same time. Several other modifications were needed, especially when executing the methods on the Jetson development kits. For all Torch-based methods, unless otherwise noted, we used Python 3.6 and PyTorch 1.6.0 with TorchVision 0.7.0.

We label each team's method by the team name they used in the challenge (or a shortened version). DiDi Chuxing (DiDi AI Labs) had two separate teams in the challenge, under the names *DiDiMapVision* and *Orange-Control*, which we will name *DiDi-MV* and *DiDi-OC*, respectively.

DiDi-OC. The team's code[1] is based on PyTorch, which we installed on servers within an Anaconda virtual environment. On Jetson devices, we used Python virtual environments instead, relying on the pre-installed JetPack libraries to simplify installation.

DiDi-MV [3]. The second DiDi team also had code[2] that depended on Torch, but requires PyTorch 1.1 and TorchVision 0.3, which we were able to install and run on Ubuntu servers via Anaconda but not on the CentOS server, due to an unknown run-time error we could not easily trace. Our solution for running the code on the Tesla V100 system required building an Ubuntu-based docker container which we executed in the CentOS 7 environment of that system. The code also requires the *MMdetection* package, which we could not get properly working on the Jetson units, along with PyTorch 1.1, preventing us from evaluating this code on edge devices.

ENGIE [53]. The ENGIE code[3] relies on PyTorch and the *nvidia-dali* package, which we were able to install on all servers via Anaconda. The *nvidia-dali* package, which was used for its video input and pipelining abilities, is not currently available for the ARM architecture. Therefore, when running the code on the Jetson devices, we replaced *nvidia-dali* code with sequential executions that used the *sk-video* package for reading in frames.

[1] DiDi-OC: https://github.com/liwenwei123/AIC_2020_Challenge_Track-1
[2] DiDi-MV: https://github.com/Jilliansea/DTTM-Vehicle-Counting
[3] ENGIE: https://github.com/AndresOsp/Track1

CMU [71]. The CMU code[4] was easily installed via Anaconda on servers. While installing the requisite packages on the Jetson development kit ARM architecture was tedious, it eventually succeeded. However, we encountered many execution stops on the Jetson devices, where the program continued to execute without reporting any error but no frames being processed either on the GPU or the CPU. These constant delays, coupled with the slow execution pace of the method on the Jetson devices, prevented us from finishing processing all videos on any of these devices. After over a week of constant method restarts, we were able to process 22 videos on the Jetson TX2, 15 videos on the Jetson NX, and 6 videos on the Jetson Nano. As a result, this method will not be included in some of the edge device evaluations that require results from all videos.

Baidu [44]. The Baidu code[5] was provided as a docker container running a CentOS 7 environment with PaddlePaddle 1.7.0 and CUDA 9.0. The environment ran successfully on all our servers but could not be executed on the Jetson units and PaddlePaddle is not yet compatible with the ARM CPU architecture.

5 Evaluation Results

We start our evaluation by comparing the results of the five methods in our evaluation versus the official results for Track 1 of the 4th AI City Challenge. Our results are quite different, which can be explained by the change in evaluation methodology. We then dig deeper into the factors that contribute to each method's success. Finally, after analyzing the correlation between different performance factors when the methods are executed on a server, we turn our attention to the ability of the methods to execute in real-time on edge devices.

5.1 Comparison to Challenge Results

The 4th AI City Challenge crowned the Baidu team as the winner of Track 1 and the DiDi-MV team as runner-up, based on the results shown in the top portion of Table 2. The bottom rows in the table show the results of our current analysis on the *Titan RTX* server, our most performant system among those we have tested. Team names and $S1$ scores of the winner are bold and of the runner-up are in italics. At first glance, it seems there is a major upset in the team rankings with the Baidu team dropping to the lowest spot and the two DiDi teams swapping places. We remind the reader, however, that our current evaluation methodology differs than the official challenge result in several major ways, which ultimately makes the comparison of the two rankings irrelevant. First, the challenge results were obtained on Dataset B, a very similar though slightly shorter version of Dataset A, which was used in our experiments. Second, the challenge allowed teams to process multiple videos at a time, while we enforced separate executions of the methods on each of the 31 videos in the dataset. This likely had a major impact on the Baidu method which relies heavily on GPU multiprocessing and uses a large custom-built model which had to be loaded repeatedly for each video processing, while it helped methods that relied on more compact models.

[4] CMU: https://github.com/Lijun-Yu/zero_virus

[5] Baidu: https://github.com/PaddlePaddle/Research/tree/master/CV/VehicleCounting

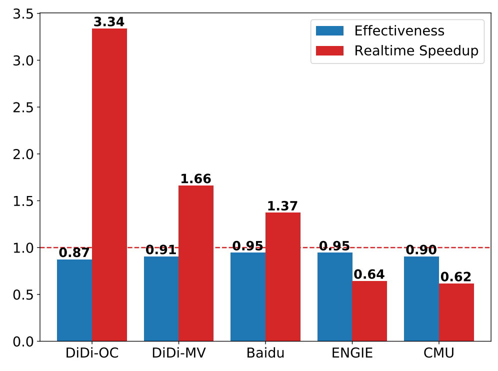
Table 2 Overall Evaluation Results

Dataset	Team	$S1_{\text{effect}}$	$S1_{\text{effic}}$	S1
B	**Baidu**	0.94588	0.85423	**0.91839**
B	*DiDi-MV*	0.90511	0.87973	*0.89750*
B	DiDi-OC	0.87457	0.94009	0.89422
B	ENGIE	0.94568	0.68890	0.86865
B	CMU	0.90425	0.67530	0.83557
A	**DiDi-OC**	0.87187	0.95104	**0.89562**
A	*DiDi-MV*	0.91051	0.82729	*0.88554*
A	ENGIE	0.92922	0.71189	0.86403
A	CMU	0.91195	0.66357	0.83743
A	Baidu	0.93829	0.34877	0.76144

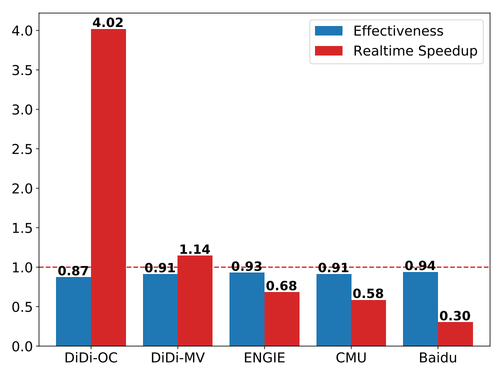
Fig. 4 compares the methods with respect to their effectiveness scores and real-time execution potential. The blue bars showcase the method effectiveness, which takes values in the range $[0, 1]$ (higher is better). The red bars show speedup (or slowdown) of the methods with respect to the video run-time (higher is better), which is depicted by the horizontal red dashed line. While having the lowest effectiveness score among the methods under comparison, DiDi-OC is able to greatly outperform all other methods with respect to efficiency, achieving 3.34× and 4.02× speedup over real-time execution in the challenge and the current experiment, respectively. Two other methods were able to achieve real-time efficiency in the challenge, DiDi-MV and Baidu, but only DiDi-MV was also able to run in real-time in our evaluation.

5.2 Execution Factors

In order to better understand the factors that played a role in the method execution, we analyzed each method's CPU, GPU, and resident memory utilization on each of



(a)



(b)

Fig. 4 Real-time execution speedup and effectiveness comparisons of the five methods in the challenge (a) and the current (b) evaluation. Best viewed in color.

Table 3 Method Execution Factors

Server	Method	%CPU	%GPU	Mem (GB)
Tesla V100	DiDi-OC	193	34	3.03
Tesla V100	DiDi-MV	6514	25	7.24
Tesla V100	ENGIE	51	48	2.12
Tesla V100	CMU	1263	95	2.35
Tesla V100	Baidu	188	20	3.83
Titan RTX	DiDi-OC	168	49	2.89
Titan RTX	DiDi-MV	2144	24	7.68
Titan RTX	ENGIE	71	55	2.28
Titan RTX	CMU	461	85	2.47
Titan RTX	Baidu	122	25	2.75
Titan X	DiDi-OC	124	55	2.58
Titan X	DiDi-MV	1502	26	6.69
Titan X	ENGIE	67	65	1.86
Titan X	CMU	296	40	2.12
Titan X	Baidu	74	35	2.21
Titan Xp	DiDi-OC	125	60	2.58
Titan Xp	DiDi-MV	1495	25	6.71
Titan Xp	ENGIE	64	65	1.84
Titan Xp	CMU	318	45	2.12
Titan Xp	Baidu	88	32	2.21

the four servers. The values reported in each row of Table 3 are averaged over executions for all videos in the dataset. The %CPU and Memory (Mem) values were obtained from the "Percent of CPU this job got" and "Maximum resident set size" values provided by the GPU *time* utility. The tool reports aggregate CPU utilization across all threads associated with a program, thus resulting in values higher than 100% for multi-threaded program executions. The %GPU values were obtained from repeated observations of GPU utilization using the $nvidia - smi$ tool.

The results indicate that DiDi-MV was the only program to effectively use CPU multi-threading as a tool to improve execution efficiency. However, even given 81.4% saturation of all threads available in the Tesla V100 system, the method was unable to keep the GPU busy with work. This indicates the method could be improved by allocating more of the workload to the GPU or somehow reducing the work assigned to the CPU threads.

While the CPU thread saturation is well below 20% for the CMU method, indicating a potential load imbalance problem, it is the only method that could nearly saturate the GPU on systems with powerful CPUs (Tesla V100 and Titan RTX). Overall, all models use less than 8GB RAM for their execution on one video, though they may require much more RAM if allowed to execute over multiple videos at the same time.

5.3 Effectiveness Factors

Effectiveness plays a big factor in the success of the counting methods. Ideally, the methods would have close to 100% effectiveness. While the majority of the methods obtain higher than 91% effectiveness, the improvement in effectiveness seems to come at the cost of efficiency. To better understand each methods' execution factors, we computed each method's effectiveness in different illumination and weather scenarios. Fig. 5 shows these results for *all* videos as well as the *day*, *dawn*, *rain*, and *snow* subsets of videos.

The model with the highest effectiveness, Baidu, is well adapted to tracking vehicles in diverse weather scenarios. While other models struggle with rain or snow scenes, Baidu shows a strong performance in all video types. Of the three typically problematic scenarios, dawn, rain, and snow, ENGIE struggles with one, CMU and DiDi-MV struggle with two, and DiDi-OC struggles with all three scenarios. This indicates that additional work is needed to make these models robust to weather and illumination changes without incurring additional execution penalties.

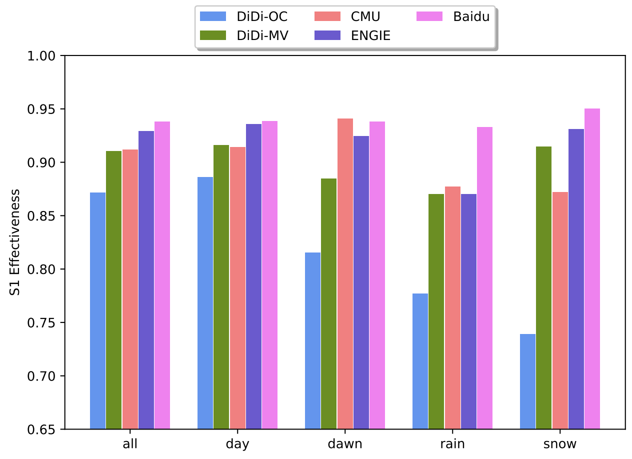


Fig. 5 Effectiveness of each method in different illumination and weather scenarios. Best viewed in color.

5.4 Efficiency Factors

One important factor in execution efficiency is the hardware (CPU and GPU) available to run the program. In our experiments, in order to showcase how different hardware can influence the model inference efficiency, we tested each model on four powerful servers with different GPUs: Tesla V100, Titan X, Titan Xp, and Titan RTX.

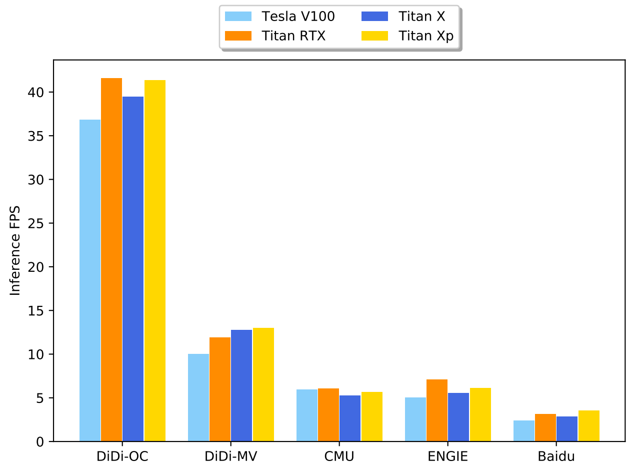


Fig. 6 Efficiency of each method across all servers. Best viewed in color.

Fig. 6 showcases the inference efficiency of each method on each of the four servers in terms of the number of frames each system is able to process every second (Inference FPS). Interestingly, all servers exhibit similar efficiency performance, with only small variations in each method's performance across servers. Somewhat surprisingly, however, the stat-of-the-art Tesla V100 system was outperformed by more commodity servers with Titan Xp and Titan RTX cards. While the NVIDIA V100 excels at training models and would outperform the other servers in those tasks, it is comparable with the Titan cards for inference tasks.

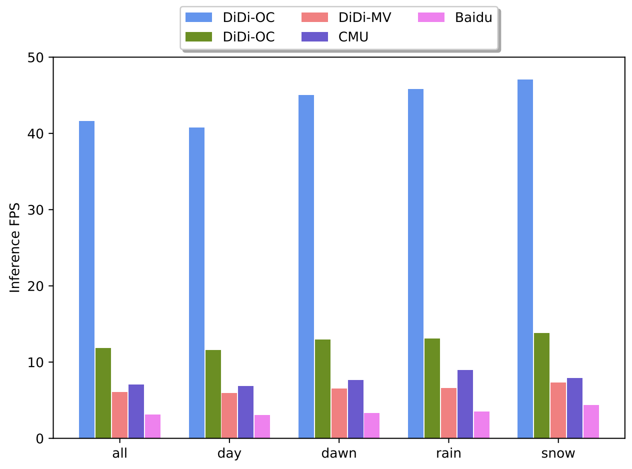


Fig. 7 Inference efficiency for each method on the Titan RTX server in different illumination and weather scenarios. Best viewed in color.

An important question is whether certain videos with "harder" to solve scenarios would negatively affect execution efficiency. Fig. 7 shows the inference efficiency for each method on the Titan RTX server in different illumination and weather scenarios. The data indicate an opposite effect. As methods miss vehicle tracking details, which result in lower counting effectiveness scores, execution efficiency tends to increase. This may be due to a failure of the underlying detector to identify all cars in the scene in each frame.

5.5 Factor correlation

In addition to the relationship between efficiency and effectiveness, we are interested in discovering whether a relationship exists between the number of vehicles or the number of movements in a video and either efficiency or effectiveness. Towards that goal, we computed the correlation matrix between these variables, which is shown visually in Fig. 8. The figure shows the distribution of each variable on the matrix diagonal and uses dots and bars of different colors corresponding to the illumination and weather type of video. Additionally, Table 4 shows the Pearson correlation coefficients between efficiency, effectiveness, and the number of movements of interest (# MOI) and the number of vehicles (# Vehicles). Significant correlations with *p-values* below 0.05 are marked in bold. We note that, for all methods, the correlation between # MOI and # Vehicles was roughly 0.37, with a *p-value* of 0.01801, showing

Table 4 Pearson Correlation of Factors

	Efficiency Correlation		
Method	Effectiveness	# MOI	# Vehicles
DiDi-OC	-0.2328	0.1544	0.2458
DiDi-MV	-0.0765	-0.0889	-0.3091
ENGIE	-0.3061	**0.3552**	**0.4043**
CMU	-0.1948	0.1721	0.1826
Baidu	0.2842	-0.0935	-0.2366

	Effectiveness Correlation		
Method	Efficiency	# MOI	# Vehicles
DiDi-OC	-0.2328	-0.3178	0.1489
DiDi-MV	-0.0765	-0.1713	0.1362
ENGIE	-0.3061	-0.2851	-0.0004
CMU	-0.1948	-0.1431	0.0063
Baidu	0.2842	-0.3479	-0.1210

that, as expected, an increase in the number of traffic lanes or movements of interest is in general associated with an increase in the number of encountered vehicles on the road.

The data provide some interesting results. First of all, an increase in the number of vehicles does not seem to affect the effectiveness of the methods considerably. In particular, the # Vehicles and Effectiveness variables are entirely not correlated for ENGIE and CMU. Interestingly, the number of vehicles and of MOIs are also not strongly correlated with efficiency for most methods, with the exception of ENGIE. It is possible that, as the number of vehicles grows, so does the number of *active tracks* whose next position should be estimated based on their motion model, which could slow down the tracking step in this method.

5.6 Edge Computing

An additional goal in our study was to ascertain the readiness of the state-of-the-art methods to be deployed in the field, potentially running on edge devices connected directly to the traffic cameras. Towards that end, we evaluated the methods on 3 different NVIDIA Jetson devices, including the TX2, NX, and Nano models, the details of which can be found in Table 1. Given several setbacks encountered during our experiment, the details of which can be found in Section 4.4, we could only successfully run DiDi-OC and ENGIE on all Jetson devices and all videos and CMU on some of the videos.

Fig. 9 shows the inference efficiency of the executed methods on the three Jetson edge devices, while Fig. 10 shows the per video inference frame rate on the Jetson NX device for the two methods we could execute for all videos. It is clear that devices with more powerful GPUs have an advantage when executing these methods. Unlike the server-based % GPU utilization results in Table 3, all methods utilized the Jetson GPU almost constantly at a rate of 100%. One problem area for CMU and ENGIE was the limited RAM available on the Jetson devices. Both methods ran out of RAM and heavily utilized the swap memory space, which we had increased in size to 24 GB. While the availability of the swap memory allowed the methods to complete their execution, the added burden of swapping pages between the disk and RAM participated in the methods' poor performance on the Jetson devices.

6 Technology Readiness

The 4th AI City Challenge has shown that several methods already exist that can achieve good performance, from both the efficiency and effectiveness points of view, for the problem of multi-class multi-movement vehicle counting. High effectiveness, however, comes at the cost of bulky models that impede efficiency. The winning methods in the challenge mitigated this problem by relying heavily on multiprocessing to keep available GPUs constantly busy. In our study, we have shown that the top methods, including DiDi-MV, ENGIE, and Baidu, are effective not only in normal daylight, but also in less desirable scenarios such as dawn, rain, or snow. Given appropriate hardware capabilities (e.g., NVIDIA Titan RTX or better), the methods are also capable of real-time processing of video streams.

An additional question we wanted to answer was whether the state-of-the-art methods were capable of being deployed on the edge, potentially on the same pole as the traffic camera. It is clear when looking at Fig. 9 that, given that the average frame rate of the processed videos was 10 fps, the Jetson devices were not able to process the streams in real-time. Only DiDi-OC was somewhat close on the Jetson NX, processing the stream at roughly 0.5x real-time. We must point out that the models employed by these methods have not been optimized for edge device execution, via techniques like quantization, which is an area of future work needed in this field. An important question that will need to be answered is whether efficiency improvements that may allow real-time inference on edge devices will be possible without any or with only little loss of counting effectiveness.

The work done thus far to advance the state-of-the-art for multi-class multi-movement vehicle counting shows that the methods are on the verge of exceeding the capabilities needed for real-world deployment. Continuing technological advancements, such as the recently released Jetson AGX Xavier edge device, will

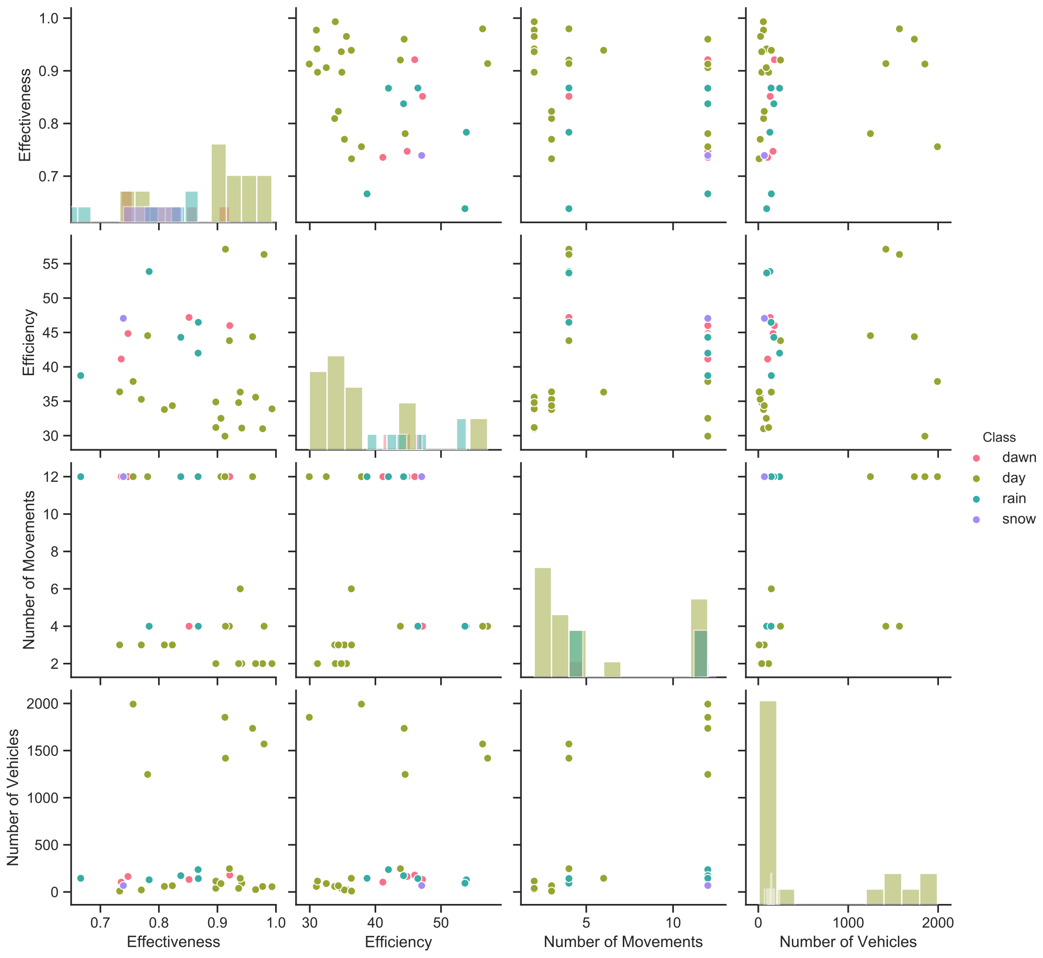


Fig. 8 Correlation of several factors that may affect execution efficiency and effectiveness for the DiDi-OC algorithm on the Titan-RTX system.

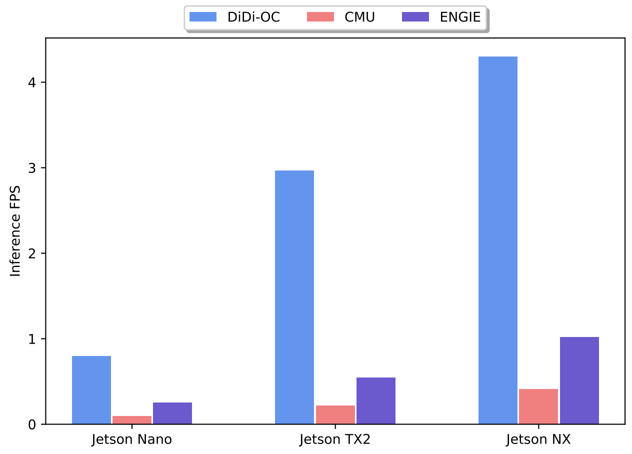


Fig. 9 Inference efficiency for each method executed on three Jetson edge devices.

allow existing methods to execute more efficiently. At the same time, researchers are encouraged to continuously improve their algorithms, both with respect to their robustness to unusual scenarios, but also to their ability to execute efficiently on edge devices. Success

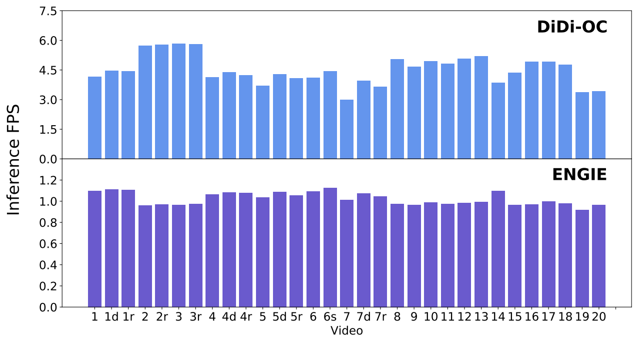


Fig. 10 Per video inference efficiency for each method executed on the Jetson NX edge device.

in these areas will enable ITSs that leverage existing traffic camera infrastructure without the need of costly in-road sensors.

7 Conclusion

In this article, we discussed the problem of video-based multi-class multi-movement vehicle counting, its importance in future intelligent transportation systems, and the current state-of-the-art for solving this problem. In order to ascertain the readiness of the current technology for real-world deployment, we executed a series of comprehensive experiments, using traffic footage with $O(5)$ vehicles captured from 20 different vantage points and covering various lighting and weather conditions. Several promising approaches were identified that can effectively count vehicles in diverse weather scenarios. While the methods cannot easily be executed in real-time on Jetson edge devices, they show much promise to achieve this capability in the near future.

Acknowledgements We gratefully acknowledge the support of the NVIDIA Corporation with the donation of two NVIDIA TITAN Xp GPUs and two NVIDIA Jetson Nano development kits for our research.

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. Abdelhalim, A., Abbas, M.: Towards real-time traffic movement count and trajectory reconstruction using virtual traffic lanes. In: Proc. CVPR Workshops. Seattle, WA, USA (2020) 1, 5
2. Arinaldi, A., Pradana, J.A., Gurusinga, A.A.: Detection and classification of vehicles for traffic video analytics. Procedia Computer Science **144**, 259 – 268 (2018). DOI https://doi.org/10.1016/j.procs.2018.10.527. INNS Conference on Big Data and Deep Learning 4
3. Bai, B., Xu, P., Xing, T., Wang, Z.: A robust trajectory modeling algorithm for traffic flow statistic. In: Proc. CVPR Workshops. Seattle, WA, USA (2020) 1, 5, 8
4. Bai, S., He, Z., Lei, Y., Wu, W., Zhu, C., Sun, M.: Traffic anomaly detection via perspective map based on spatial-temporal information matrix. In: Proc. CVPR Workshops (2019) 1
5. Bergmann, P., Meinhardt, T., Leal-Taixe, L.: Tracking without bells and whistles (2019) 4
6. Bewley, A., Ge, Z., Ott, L., Ramos, F., Upcroft, B.: Simple online and realtime tracking (2016) 5
7. Bhandary, N., MacKay, C., Richards, A., Tong, J., Anastasiu, D.C.: Robust classification of city roadway objects for traffic related applications. In: 2017 IEEE Smart World NVIDIA AI City Challenge, SmartWorld'17. IEEE, Piscataway, NJ, USA (2017) 1, 4
8. Bochinski, E., Eiselein, V., Sikora, T.: 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. Lecce, Italy (2017). URL http://elvera.nue.tu-berlin.de/files/1517Bochinski2017.pdf 4
9. Bochinski, E., Senst, T., Sikora, T.: Extending iou based multi-object tracking by visual information. In: IEEE International Conference on Advanced Video and Signals-based Surveillance, pp. 441–446. Auckland, New Zealand (2018). URL http://elvera.nue.tu-berlin.de/files/1547Bochinski2018.pdf 4
10. Bui, N.K.H., Yi, H., Cho, J.: A vehicle counts by class framework using distinguished regions tracking at multiple intersections. In: Proc. CVPR Workshops. Seattle, WA, USA (2020) 1, 5
11. Chang, M.C., Chiang, C.K., Tsai, C.M., kai Chang, Y., Chiang, H.L., Wang, Y.A., Chang, S.Y., Li, Y.L., Tsai, M.S., Tseng, H.Y.: AI City Challenge 2020 – Computer vision for smart transportation applications. In: Proc. CVPR Workshops. Seattle, WA, USA (2020) 1, 5
12. Chen, K., Gong, S., Xiang, T., Loy, C.C.: Cumulative attribute space for age and crowd density estimation. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition, pp. 2467–2474 (2013) 3
13. Chen, T.S., Lee, M.Y., Liu, C.T., Chien, S.Y.: Viewpoint-aware channel-wise attentive network for vehicle re-identification. In: Proc. CVPR Workshops. Seattle, WA, USA (2020) 1
14. Chen, Y., Hu, W.: Robust vehicle detection and counting algorithm adapted to complex traffic environments with sudden illumination changes and shadows. Sensors (Basel, Switzerland) **20**(9), 2686 (2020). DOI 10.3390/s20092686. URL https://pubmed.ncbi.nlm.nih.gov/32397207 4
15. Chen, Y., Jing, L., Vahdani, E., Zhang, L., Tian, Y., He, M.: Multi-camera vehicle tracking and re-identification on AI City Challenge 2019. In: Proc. CVPR Workshops (2019) 1
16. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to algorithms. The MIT Press (2009) 5
17. Doshi, K., Yilmaz, Y.: Fast unsupervised anomaly detection in traffic videos. In: Proc. CVPR Workshops. Seattle, WA, USA (2020) 1
18. Eckstein, V., Schumann, A.: Large scale vehicle re-identification by knowledge transfer from simulated data and temporal attention. In: Proc. CVPR Workshops. Seattle, WA, USA (2020) 1
19. Feng, W., Ji, D., Wang, Y., Chang, S., Ren, H., Gan, W.: Challenges on large scale surveillance video analysis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (2018) 1
20. Ghiasi, G., Lin, T., Le, Q.V.: NAS-FPN: learning scalable feature pyramid architecture for object detection. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019, pp. 7036–7045. Computer Vision Foundation / IEEE (2019). DOI 10.1109/CVPR.2019.00720 4
21. Ghiasi, G., Lin, T.Y., Le, Q.V.: Nas-fpn: Learning scaleable feature pyramid architecture for object detection. Proceedings of the IEEE Confrence on Computer Vision and Pattern Recognition pp. 7036–7045 (2019) 5
22. Girshick, R.: Fast r-cnn. In: International Conference on Computer Vision (ICCV) (2015) 4
23. Guerrero-Ibáñez, J., Zeadally, S., Contreras-Castillo, J.: Sensor technologies for intelligent transportation systems. Sensors (Basel, Switzerland) **18**(4), 1212 (2018). DOI 10.3390/s18041212. URL https://pubmed.ncbi.nlm.nih.gov/29659524 3
24. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: ICCV, pp. 2961–2969 (2017) 5
25. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn (2017) 4
26. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition (2015) 6

27. He, S., Luo, H., Chen, W., Zhang, M., Zhang, Y., Wang, F., Li, H., Jiang, W.: Multi-domain learning and identity mining for vehicle re-identification. In: Proc. CVPR Workshops. Seattle, WA, USA (2020) 1
28. He, Z., Lei, Y., Bai, S., Wu, W.: Multi-camera vehicle tracking with powerful visual features and spatial-temporal cue. In: Proc. CVPR Workshops (2019) 1
29. Hou, Y., Du, H., Zheng, L.: Think small, deliver big: A locality aware city-scale multi-camera vehicle tracking system. In: Proc. CVPR Workshops (2019) 1
30. Hsu, H.M., Huang, T.W., Wang, G., Cai, J., Lei, Z., Hwang, J.N.: Multi-camera tracking of vehicles based on deep features Re-ID and trajectory-based camera link modelsodels. In: Proc. CVPR Workshops (2019) 1
31. Hua, S., Anastasiu, D.C.: Effective vehicle tracking algorithm for smart traffic networks. In: Thirteenth IEEE International Conference on Service-Oriented System Engineering (SOSE), SOSE 2019. IEEE (2019) 4
32. Hua, S., Kapoor, M., Anastasiu, D.C.: Vehicle tracking and speed estimation from traffic videos. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPRW'18. IEEE (2018) 4
33. Ignatius Moses Setiadi, D.R., Fratama, R.R., Ayu Partiningsih, N.D., Rachmawanto, E.H., Sari, C.A., Andono, P.N.: Real-time multiple vehicle counter using background subtraction for traffic monitoring system. In: 2019 International Seminar on Application for Technology of Information and Communication (iSemantic), pp. 1–5 (2019) 3
34. Leclercq, L., Wang, Y., Yang, X., Liang, H., Liu, Y.: A review of the self-adaptive traffic signal control system based on future traffic environment. Journal of Advanced Transportation **2018**, 1096123 (2018). DOI 10.1155/2018/1096123. URL https://doi.org/10.1155/2018/1096123 1, 3
35. Lee, S., Park, E., Yi, H., Lee, S.H.: StRDAN: Synthetic-to-real domain adaptation network for vehicle re-identification. In: Proc. CVPR Workshops. Seattle, WA, USA (2020) 1
36. Li, P., Li, G., Yan, Z., Li, Y., Lu, M., Xu, P., Gu, Y., Bai, B.: Spatio-temporal consistency and hierarchical matching for multi-target multi-camera vehicle tracking. In: Proc. CVPR Workshops (2019) 1
37. Li, Y., Wu, J., Bai, X., Yang, X., Tan, X., Li, G., Wen, S., Zhang, H., Ding, E.: Multi-granularity tracking with modularlized components for unsupervised vehicles anomaly detection. In: Proc. CVPR Workshops. Seattle, WA, USA (2020) 1
38. Liang, M., Huang, X., Chen, C., Chen, X., Tokuta, A.: Counting and classification of highway vehicles by regression analysis. IEEE Transactions on Intelligent Transportation Systems **16**(5), 2878–2888 (2015) 3
39. Lin, T.Y., Dollar, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. The IEEE Confrenece on computer Vision and Pattern Recognition (CVPR) (2017) 6
40. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection (2018) 5
41. Lin, T.Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C.L., Dollár, P.: Microsoft coco: Common objects in context (2015) 5
42. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. Lecture Notes in Computer Science p. 21–37 (2016). DOI 10.1007/978-3-319-46448-0_2. URL http://dx.doi.org/10.1007/978-3-319-46448-0_2 4
43. Liu, X., Wang, Z., Feng, J., Xi, H.: Highway vehicle counting in compressed domain. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3016–3024 (2016) 3
44. Liu, Z., Zhang, W., Gao, X., Meng, H., Xue, Z., Tan, X., Zhu, X., Zhang, H., Wen, S., Ding, E.: Robust movement-specific vehicle counting at crowded intersections. In: Proc. CVPR Workshops. Seattle, WA, USA (2020) 1, 5, 6, 9
45. Mirthubashini, J., Santhi, V.: Video based vehicle counting using deep learning algorithms. In: 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), pp. 142–147 (2020) 4
46. Nagaraj, S., Muthiyan, B., Ravi, S., Menezes, V., Kapoor, K., Jeon, H.: Edge-based street object detection. In: 2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), pp. 1–4 (2017) 1
47. Nam, H., Baek, M., Han, B.: Modeling and propagating cnns in a tree structure for visual tracking (2016) 4
48. Nam, H., Han, B.: Learning multi-domain convolutional neural networks for visual tracking. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016) 4
49. Naphade, M., Anastasiu, D.C., Sharma, A., Jagrlamudi, V., Jeon, H., Liu, K., Chang, M.C., Lyu, S., Gao, Z.: The NVIDIA AI City Challenge. In: Prof. SmartWorld. Santa Clara, CA, USA (2017) 1, 2, 4
50. Naphade, M., Chang, M.C., Sharma, A., Anastasiu, D.C., Jagarlamudi, V., Chakraborty, P., Huang, T., Wang, S., Liu, M.Y., Chellappa, R., Hwang, J.N., Lyu, S.: The 2018 NVIDIA AI City Challenge. In: Proc. CVPR Workshops, pp. 53—60 (2018) 1, 2
51. Naphade, M., Tang, Z., Chang, M.C., Anastasiu, D.C., Sharma, A., Chellappa, R., Wang, S., Chakraborty, P., Huang, T., Hwang, J.N., Lyu, S.: The 2019 ai city challenge. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, pp. 452—460 (2019) 1, 2
52. Naphade, M., Wang, S., Anastasiu, D.C., Tang, Z., Chang, M.C., Yang, X., Zheng, L., Sharma, A., Chellappa, R., Chakraborty, P.: The 4th ai city challenge. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, pp. 2665—2674 (2020) 1, 2, 4, 5, 7
53. Ospina, A., Torres, F.: Countor: Count without bells and whistles. In: Proc. CVPR Workshops. Seattle, WA, USA (2020) 1, 5, 8
54. Redmon, J., Farhadi, A.: YOLOv3: An incremental improvement. arXiv:1804.02767 (2018) 4, 5
55. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems, pp. 91–99 (2015) 4, 5, 6
56. Rosas-Arias, L., Portillo-Portillo, J., Hernandez-Suarez, A., Olivares-Mercado, J., Sanchez-Perez, G., Toscano-Medina, K., Perez-Meana, H., Sandoval Orozco, A.L., García Villalba, L.J.: Vehicle counting in video sequences: An incremental subspace learning approach. Sensors (Basel, Switzerland) **19**(13), 2848 (2019). DOI 10.3390/s19132848. URL https://pubmed.ncbi.nlm.nih.gov/31252574 4

57. Shi, H., Liu, Z., Fan, Y., Wang, X., Huang, T.: Effective object detection from traffic camera videos. In: 2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), pp. 1–5 (2017) 1
58. Shine, L., A, V.M., V, J.C.: Fractional data distillation model for anomaly detection in traffic videos. In: Proc. CVPR Workshops. Seattle, WA, USA (2020) 1
59. Špaňhel, J., Herout, A., Bartl, V., Folenta, J.: Determining vehicle turn counts at multiple intersections by separated vehicle classes using CNNs. In: Proc. CVPR Workshops. Seattle, WA, USA (2020) 1, 5
60. Supreeth, H.S.G., Patil, C.M.: Moving object detection and tracking using deep learning neural network and correlation filter. In: 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), pp. 1775–1780 (2018) 4
61. Tan, M., Pang, R., Le, Q.V.: Efficientdet: Scalable and efficient object detection (2019) 4
62. Tang, Z., Hwang, J.: Moana: An online learned adaptive appearance model for robust multiple object tracking in 3d. IEEE Access **7**, 31934–31945 (2019) 4
63. Tang, Z., Naphade, M., Liu, M.Y., Yang, X., Birchfield, S., Wang, S., Kumar, R., Anastasiu, D., Hwang, J.N.: Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019) 2
64. Tang, Z., Wang, G., Xiao, H., Zheng, A., Hwang, J.: Single-camera and inter-camera vehicle tracking and 3d speed estimation based on fusion of visual and semantic features. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 108–1087 (2018) 1, 4
65. Tran, M.T., Nguyen, T.V., Le, T.N., Nguyen, K.T., Dinh, D.T., Nguyen, T.A., Nguyen, H.D., Nguyen, T.T., Hoang, X.N., Vo-Ho, V.K., Do, T.L., Nguyen, L., Le, M.Q., Nguyen-Dinh, H.P., Pham, T.T., Nguyen, E.R., Tran, Q.C., Vu-Le, T.A., Nguyen, T.P., Nguyen, X.V., Tran, V.H., Dao, H., Nguyen, Q.T., Tran, M.K., Diep, G.H., Do, M.: iTASK - Intelligent traffic analysis software kit. In: Proc. CVPR Workshops. Seattle, WA, USA (2020) 1, 5
66. Wang, Z., Zheng, L., Liu, Y., Li, Y., Wang, S.: Towards real-time multi-object tracking. 2020 European Conference on Computer Vision (2019) 5
67. Wen, L., Du, D., Cai, Z., Lei, Z., Chang, M., Qi, H., Lim, J., Yang, M., Lyu, S.: UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking. Computer Vision and Image Understanding (2020) 1
68. Wojke, N., Bewley, A., Paulus, D.: Simple online and realtime tracking with a deep association metric. In: Proc. ICIP, pp. 3645–3649 (2017) 4
69. Wojke, N., Bewley, A., Paulus, D.: Simple online and realtime tracking with a deep association metric. In: ICIP, pp. 3645–3649 (2017) 5, 6
70. Wu, M., Zhang, G., Bi, N., Xie, L., Hu, Y., Gao, S., Shi, Z.: Multiview vehicle tracking by graph matching model. In: Proc. CVPR Workshops (2019) 1
71. Yu, L., Feng, Q., Qian, Y., Liu, W., Hauptmann, A.: Zero-VIRUS: Zero-shot vehicle route understanding system for intelligent transportation. In: Proc. CVPR Workshops. Seattle, WA, USA (2020) 5, 9
72. Zheng, Z., Jiang, M., Wang, Z., Wang, J., Bai, Z., Zhang, X., Yu, X., Tan, X., Yang, Y., Wen, S., Ding, E.: Going beyond real data: A robust visual representation for vehicle re-identification. In: Proc. CVPR Workshops. Seattle, WA, USA (2020) 1
73. Zhou, X., Wang, D., Krähenbühl, P.: Objects as points. In: arXiv preprint arXiv:1904.07850 (2019) 4
74. Zhu, X., Luo, Z., Fu, P., Ji, X.: VOC-ReID: Vehicle re-identification based on vehicle-orientation-camera. In: Proc. CVPR Workshops. Seattle, WA, USA (2020) 1