An Efficient Data Transmission Framework for Connected Vehicles

Yichen Luo*, Yongtao Yao†, Junzhou Chen*, Sidi Lu* and Weisong Shi†
*Department of Computer Science, William & Mary, Williamsburg, VA 23185, USA
†Department of Computer & Information Sciences, University of Delaware, Newark, DE 19716, USA
{yluo11, jchen57, sidi}@wm.edu, {yongtao, weisong}@udel.edu

***Abstract*—Connected vehicles (CVs) face significant challenges in continuous big data transmission, resulting in high transmission bandwidth costs and impacting real-time decision-making. To address this, we propose two dynamic, driving-aware compression mechanisms based on reinforcement learning and temporal compressive sensing to intelligently compress video data. These mechanisms adapt to driving conditions, reducing bandwidth while preserving sufficient information for accurate applications such as object detection and ensuring high-quality reconstruction when needed. We also implement a Vehicle-EdgeServer-Cloud (VEC) closed-loop framework that integrates these mechanisms. Specifically, a lightweight vehicle model performs real-time detection on compressed data (measurements), while the EdgeServer receives measurements and reconstructs scenes if needed. The measurements, reconstructed video, and analysis results are then sent to the cloud for vehicle model updates. Unlike conventional methods, our framework seamlessly adapts across vehicles, EdgeServers, and the cloud, supporting efficient data transmission and dynamic model updates. Extensive evaluations were conducted on our designed roadside unit platform and robotic vehicle, both equipped with industry-grade sensors and computing units. The results demonstrate an 18× reduction in bandwidth at 320KB/s while maintaining high detection accuracy and reconstruction quality compared to non-adaptive measurements, highlighting the framework's promising real-world applications for CVs.**

***Index Terms*—Connected vehicles, compressed video, reinforcement learning, temporal compressive sensing.**

I. Introduction

As vehicle computing [1] and the 5G era unfolds, connected vehicles (CVs) are swiftly transforming the automotive industry, with emerging intelligent services driving the requirements for fast data processing and low-bandwidth communication [2], [3], in the face of high data volume. Estimates predict that by 2025, up to 400 million new vehicles will be connected [4].

Astronomical transmission bandwidth. The rapid advancement of Vehicle-to-Everything (V2X) technologies enables vehicles to communicate with edge servers and cloud infrastructures, enhancing road safety and collision avoidance [5]. For instance, by leveraging V2X, GM's Super Cruise enhances the sensing capabilities of autonomous vehicles, a type of CV [6]. In addition, to enable the commercial deployment of connected and autonomous vehicles on public roads, it is mandatory to ensure continuous data transmission for model training and teleoperation (e.g., remote assistance) [1]. Hence, transmission costs can skyrocket: projections indicate that by 2025, CVs will transmit up to 10 exabytes of data per month, a volume at least 1,000 times greater than current levels [4]. A recent Guidehouse Insights report estimates that 10 million vehicles can transmit over 20 petabytes of data annually, incurring costs exceeding $1 billion [7], [8]. These escalating data transmission costs, compounded by network congestion and increased latency, present substantial challenges for CVs.

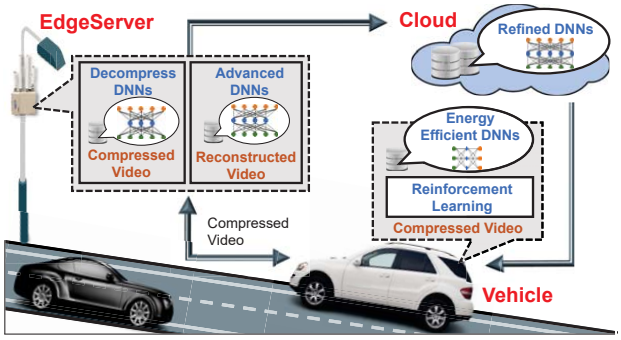

Fig. 1. A simplified illustration of the Vehicle-EdgeServer-Cloud (VEC) *closed-loop* framework. Both a cellular tower and a RSU can be considered as the EdgeServer.

Latency-sensitive automotive applications. The success of CV systems hinges on the ability to make reliable real-time decisions. According to [9], an autonomous vehicle traveling at 40km/h in an urban environment requires a response time of less than 100ms to maintain control every 1 meter. Moreover, previous experiments have shown that to proactively plan a route to avoid an obstacle 5 meters ahead, the computing latency must be under 164ms [10]. Failure to meet these latency requirements can result in significant safety risks.

Powerful and resource-hungry applications. However, today's CV frameworks are not optimized to handle the increasing volume of data and efficiently support emerging automotive intelligent services [4]. The main reason is that most neural networks focus on boosting accuracy at the expense of substantially increased model complexity [11], [12]. For example, Inception [13] and ResNet [14], which leverage deeper layers to extract hierarchical features, can reach dozens or even hundreds of layers to outperform previous networks, and a single layer may require millions of matrix multiplications. Such heavy calculation brings pressure on computing systems in terms of performance and energy consumption.

Computation-constrained vehicles. To achieve high vehicle computing speeds, a common approach is to enhance computation capability by adding more advanced computing

units. However, this approach is neither feasible nor practical for CVs as it significantly increases loading, building, and maintenance costs [1]. In the United States, the average cost to build a traditional non-luxury vehicle is roughly $30K, whereas the cost for a connected and autonomous vehicle is approximately $250K, with sensors and computing platforms accounting for almost two-thirds of the total price [1]. These high costs lead to declining consumer demand and reduced profitability for vehicle manufacturers.

In-transit Communication. Despite the rapid development of communication technologies such as Dedicated Short Range Communication (DSRC) [15], WiFi, LTE, and Cellular-Vehicle-to-Everything (C-V2X) [16], achieving ultra-fast, low-bandwidth communication for distributed and moving CVs remains challenging, particularly for video analysis [1]. *In this context, the key point is to reduce data transmission volume, which is the most direct and effective method for ultra-fast and low-bandwidth V2X communication [1].*

Bearing these concerns in mind, we design and implement a closed-loop framework (Fig. 1), the Vehicle-EdgeServer-Cloud (VEC), which incorporates EdgeServers, like roadside units (RSUs) [17], to provide efficient and stable communication. RSUs are typically deployed along roadways to enable short-range communication with vehicles, facilitating faster data transmission. Besides, given that vehicles are in motion and may experience inconsistent connectivity with the cloud, RSUs enhance communication reliability by reducing packet loss.

Our main objective is to reduce data transmission bandwidth by employing an adaptive temporal compressive sensing (TCS) technique that transmits only compressed images, referred to as measurements, for vehicle applications, while also providing an alternative and novel method to facilitate real-time computational capabilities in CVs. In this work, we choose a mainstream vehicle application, i.e., object detection, as a use case. The key contributions of this work are listed as follows:

- We design and implement a VEC closed-loop framework to integrate TCS, edge computing, and CVs. On the vehicle side, a lightweight deep learning model is tasked with performing real-time video processing (e.g., object detection) on the measurements with adaptive TCS based on the current driving environment. The EdgeServer is designed to reconstruct the original video data upon the activation of a trigger (e.g., when the number of surrounding vehicles detected exceeds three). Meanwhile, results from the EdgeServer will be sent to the cloud with the saved measurements to update the lightweight deep learning model for vehicles, ensuring continuous improvement in performance and efficiency.
- We propose and test two reinforcement learning (RL) policies to dynamically determine the optimal compression ratio (Cr) in both urban (low-speed) and highway (high-speed) scenarios (Sec. IV-A). Our experimental results demonstrate the effectiveness of adaptive TCS compared with non-adaptive TCS. In addition, we train and compare two state-of-the-art (SOTA) reconstruction models with different Cr, i.e., E2E-CNN and BIRNAT (Sec. III-E).
- The absence of experimental platforms (e.g., vehicles and RSUs), has led most research to rely on simulation-based experiments, which often fail to address real-world application requirements and testing. To address this gap, we design and develop two real-hardware platforms equipped with industry-grade computing units and sensors: i) an EdgeServer computing platform and ii) a robotic vehicle (described in Sec. III-C). These testbeds replicate real-world vehicle conditions, enabling comprehensive evaluation.
- We conduct comprehensive testing across various compression ratios ($Cr = 6, 8, 10, 15, 20$), evaluating detection accuracy (Sec. IV-A), inference latency (Sec. IV-D), transmission bandwidth (Sec. IV-E), and CPU and memory utilization (Sec. IV-F). Our proposed framework with adaptive TCS, even when constrained by limited bandwidth as low as 320KB/s, results in an $18\times$ bandwidth reduction, while maintaining as good accuracy as algorithms that process video data in the uncompressed realm.

The rest of the paper is organized as follows: related works are reviewed in Sec. II. Sec. III describes the framework design, experiment setup, and comprehensive implementation step by step. Evaluation results and discussion are presented in Sec. IV and Sec. V, respectively. Finally, Sec. VI concludes the paper.

II. Related Work

A. Background of Temporal Compressive Sensing

In video TCS, the high-speed frames of a video are modulated at a higher speed than the capture rate of the camera [18]. These modulated frames are then compressed into a single measurement. With knowledge of the modulation, multiple frames can be reconstructed from every single measurement. Fig. 2 shows the pipeline of TCS. The Cr video frames (top-left) are modulated by Cr different modulation patterns (bottom-left) and then compressed to a single compressed measurement shown in the top-right. Mathematically, let $\mathrm{X_k} \in R^{N_x \times N_y}$ denote the k-th video frame, $\forall k = 1, \ldots Cr$.

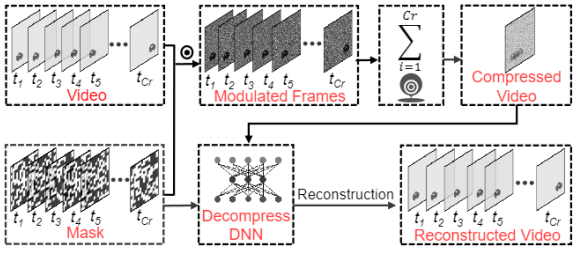

Fig. 2. The pipeline of temporal compressive sensing (TCS).

During the TCS capture, within one exposure time, each frame is modulated (element-wise product) by a unique pattern $\mathrm{X_k} \in R^{N_x \times N_y}$, $\forall k = 1, \ldots, Cr$. These frames (top-middle) are then summed to a single measurement $\mathrm{Y} \in R^{N_x \times N_y}$ captured by the detector (CCD or CMOS camera). The forward model is $\mathrm{Y} = \sum_{k=1}^{Cr} \mathrm{X_k} \odot \mathbf{M}_k + \mathbf{G}$, where $\odot$ denotes the element-wise product and $\mathbf{G}$ denotes the measurement noise. Here, the element-wise product operation of the TCS capture is computationally simple, resulting in much lower overhead and latency compared to more complex machine-learning methods.

B. Adaptive Temporal Compressive Imaging

Considering the real-world application of measurement-based object detection for CVs, a higher value of Cr is normally selected to pursue a faster inference speed at the expense of lower detection accuracy and reconstruction quality. However, in this work, we strike a balance between Cr, the driving environment, and related performance, which calls for adaptive TCS [19], [20]. Unlike previous works that conduct adaptive TCS in the spatial domain [21], [22], we achieve adaptive TCS in the temporal domain to determine the optimal Cr before compression and reconstruction. Only one paper [23] considers a similar problem, but only estimates the motion of the objects to adapt Cr with a look-up table. It also does not take account of driving scene complexity. To the best of our knowledge, this is the first work to deploy adaptive TCS on CVs for efficient data transmission under varying driving conditions (e.g., fast-motion and slow-motion scenarios).

C. Video Reconstruction

As discussed in Sec. II-A, after the TCS camera captures $\mathbf{Y}$ (bottom-right of Fig. 2), the next task is to perform reconstruction, which aims to estimate $\mathrm{X_k}$ from Y using the given measurement matrix $\mathbf{M}_k$. Various reconstruction models have been developed to predict original high-speed video frames from a single measurement, a process known as video reconstruction [24]. With the advancements in deep learning algorithms, several deep learning-based reconstruction models have been proposed [25]–[27]. Among them, E2E-CNN [28] (End-to-End Convolutional Neural Network) set a new benchmark by achieving millisecond-level reconstruction for video TCS. Recently, BIRNAT [29] (Bidirectional Recurrent Neural Networks with Adversarial Training) is introduced, bridging the gap between reconstruction quality and speed by fully exploiting the correlation between high-speed video frames to enhance reconstruction quality. While the aforementioned models demonstrate impressive performance in terms of reconstruction quality and speed, their testing is primarily conducted on general video data or simulations, rather than on vehicles or real hardware. Moremore, previous testbeds ignore the constraints of CPU and memory resources when executing these tasks. However, considering the limited computational resources available in vehicles, such hardware limitations should be taken into account for practical applications.

D. Object Detection and Tracking

1) Object Detection: Despite advances like YOLOv10 [30], YOLO models struggle to balance accuracy and speed due to CV hardware limitations. While deep-to-shallow connections enhance accuracy, they also increase inference time by up to 20% [31]. Given the computational constraints and time-sensitive requirements of CV applications, YOLOv3, which shares the Darknet-53 backbone network with YOLOv10 and incorporates multi-scale features for object detection, achieves an optimal balance between accuracy and speed [32]; thus, we select the YOLOv3 model to be executed on dynamic measurements for our case study (object detection).

2) Overview of Multi-Object Tracking: **Tracking after detection.** Tracking safety-critical targets, such as front vehicles and pedestrians, is crucial for ensuring safe driving [33]. "Tracking-after-detection" methods are widely used to solve multi-object tracking (MOT) problems [34]–[36]. Most mainstream MOT approaches [37], [38] firstly detect all targets for each frame before tracking targets through bounding box (bbox) association. Here, bbox association involves matching the detected bboxes across consecutive frames. SORT [39] analyzes object tracking results and computes the associated appearance descriptors of objects within each frame to keep tracking. Later advancements like Deep SORT [40] are proposed to reduce the number of identity switches caused by long periods of occlusions, preventing the tracking system from mistakenly assigning one object's identity to another when objects are temporarily blocked from view.

Joint detection and tracking. Recent trends in MOT tasks are to convert existing detectors into trackers, thereby integrating both tasks within the same framework [41], [42]. For example, CenterTrack [41] tracks objects as points, jointly learning detection and association, while Tracktor [42] propagates object identities through bbox regression, eliminating the need for bbox association. Moreover, methods like stacking video frames [43] and feature warping [44], improve detection accuracy and accelerate inference. Yet, these works overlook the essential consideration of latency. Only one study aligns closely with our approach, improving Deep SORT for vehicle tracking [45] but still neglecting the crucial issue of latency, which is vital for real-time applications.

E. Additional Gaps in Previous Work

Within the VEC framework, although previous studies explored computing frameworks between vehicles and the cloud [46], or between vehicles and EdgeServers, these frameworks are inherently open-loop, lacking mechanisms to mitigate a system's sensitivity to external disturbances. For example, collaborative cloud-vehicle computation systems enable vehicles to pull compressed models from the cloud for driving behavior modeling [46]. However, they overlook the "push" process from the vehicle to the cloud. Similarly, recent studies have emphasized computation offloading [47], [48]. Yet, they neglect bidirectional data transmission between vehicles and RSUs. Conversely, our solution introduces a closed-loop framework, involving the vehicle, EdgeServers (i.e., RSU), and the cloud, which considers bidirectional data transmission (measurement uplink and model downlink). This framework also incorporates adaptive Cr to accommodate dynamic vehicle scenarios, reducing bandwidth for data transmission.

From a computing systems perspective, most research often relies on simulation data due to the lack of dedicated experimental platforms. In contrast, we design and implement an RSU platform and a robotic vehicle equipped with industry-grade computing units and sensors, facilitating comprehensive testing that closely reflects real-world conditions.

Regarding communication efficiency, previous research primarily focuses on enhancing communication mechanisms

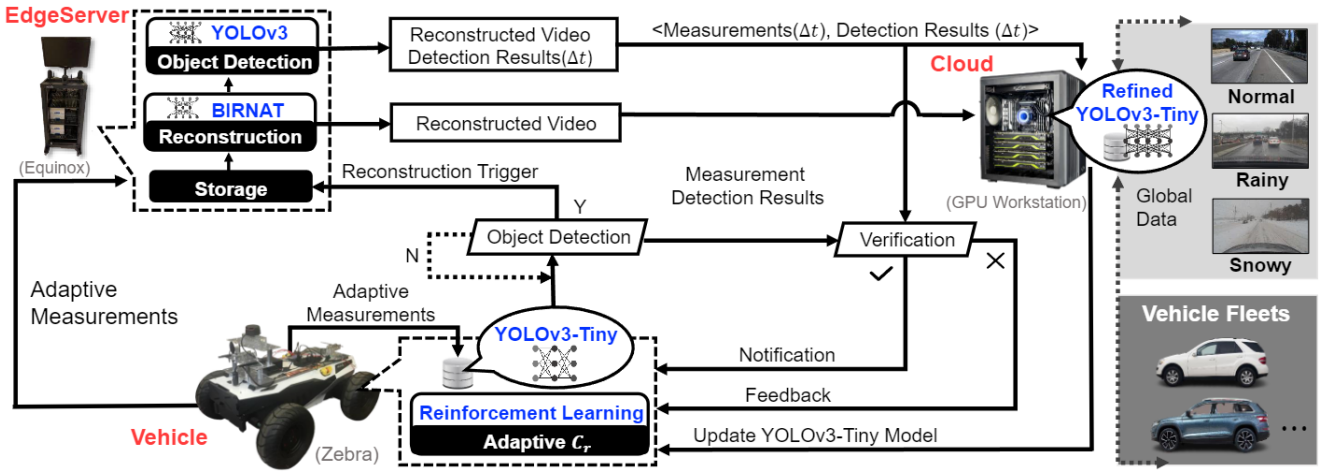

Fig. 3. The complete VEC *closed-loop* framework to integrate TCS, edge computing, and CVs. The vehicle generates (captures) adaptive measurements based on the RL, which decides adaptive Cr by involving vehicle tracking or measurement similarity analysis, and then a lightweight model (YOLOv3-Tiny) is employed to perform real-time detection based on raw measurements of vehicles. Compressed with low bandwidth measurements are also sent to the EdgeServer to save the information. When the trigger is on, the reconstruction operation is performed by BIRNAT, and a more accurate result of object detection based on the reconstructed video is described by an advanced detection network (YOLOv3), verifying the YOLOv3-Tiny's result. Meanwhile, these results will be sent to the cloud with the saved measurements to refine YOLOv3-Tiny to provide more accurate in-vehicle services. The reconstructed video will also be sent to the cloud to receive related information from vehicle fleets to conduct decision-making such as traffic control and path planning.

by developing advanced protocols [49], [50], architectures [51], and innovative communication strategies [52]–[54]. In contrast, we propose the use of adaptive TCS to improve transmission efficiency between vehicles and RSUs. To the best of our knowledge, only one related work [55] demonstrates the improvement of object detection on compressed measurements. Our work takes this technology one step further toward real applications of adaptive TCS for CVs, which is able to dynamically reduce bandwidth consumption and accelerate machine vision services.

Moreover, compared to traditional video compression standards like H.264 [56], [57] and H.265 [58], TCS is better suited for compressing data from vehicle-mounted cameras. TCS compresses data at the acquisition stage, reducing transmission bandwidth and computational delays on the vehicle side [59], [60]. It also avoids reliance on motion estimation, ensuring stable compression in dynamic scenes. TCS's simple approach lowers energy consumption [61], [62], making it ideal for low-latency, real-time applications, especially in large-scale CV deployments and complex environments.

III. Framework Design and Implementation

A. VEC Closed-loop Framework Description

As mentioned in Section I and Section II-E, the VEC closed-loop framework includes three key components. Fig. 3 provides a detailed overview of the framework process.

i) Vehicle: With the help of RL, the vehicle generates dynamic bandwidth-efficient measurements based on the adaptive compression ratio Cr. Then, an energy-efficient network, i.e., YOLOv3-Tiny [63] performs object detection (vehicle detection as an example) *directly on the measurements*, with the effectiveness of boosting inference speed.

ii) EdgeServer: The EdgeServer stores and reconstructs data from vehicles upon specific triggers, leveraging a more powerful YOLOv3 network for more accurate applications, and providing feedback or notifications to vehicles to ensure a closed-loop framework.

iii) Cloud: The cloud aggregates all useful information from the EdgeServer, such as reconstructed video detection results and saved measurements, to refine the YOLOv3-Tiny model on the vehicle for model updating purposes. Additionally, it hosts all reconstructed videos sent from vehicle fleets covering diverse driving environments (e.g., rainy, snowy, and sunny) to train related models, enabling vehicles to adapt to different weather conditions.

B. Dataset Selection

To evaluate the adaptability of our framework, we utilize extensive datasets: *i*) **Under standard driving scenarios:** The TuSimple dataset encompasses both static and dynamic objects, comprising 7,000 video clips, each with a duration of one second. The training set includes 3,626 videos alongside 3,626 annotated frames, as referenced in [64]. *ii*) **Under rainy driving scenarios:** The Berkeley Deep-Drive dataset (BDD100K) [65] is one of the largest real-world driving video datasets. It comprises 100,000 high-definition videos, representing over 1,100 hours of driving experience across various weather conditions, including rain, sunshine, and cloudy environments. *iii*) **Under snowy driving scenarios:** The Canadian Adverse Driving Conditions (CADC) dataset [66], collected during winter in the region of Waterloo, Canada, is designed to facilitate research on adverse weather conditions.

C. Phycial Testbeds

In a real-world deployment, the RSU is an example of a typical EdgeServer in our proposed VEC closed-loop framework. However, owing to the absence of experimental platforms (e.g., vehicles and RSUs), the majority of research is based either on simulated data [17], [67], [68], or conducted on commercial equipment with very limited access to real-world vehicle conditions [10], [69].

To close this gap, we design and build i) `Equinox`, an experimental RSU platform (Fig. 4(a)) and ii) `Zebra`, a robotic vehicle (Fig. 4(b)). The robotic vehicle features an Ackermann steering chassis, an industry-grade computing board, and a suite of sensors, closely simulating a real-world vehicle testbed. Consequently, the experimental outcomes from these testbeds provide practical insights that are transferable to real vehicle applications in actual driving scenarios. To be specific, `Equinox` features advanced communication, data, and computation layers, while `Zebra` is equipped with cutting-edge sensors and computing units. In addition, a powerful NVIDIA GPU Workstation (4× GeForce RTX 2080 Ti graphics cards) is working as the cloud.

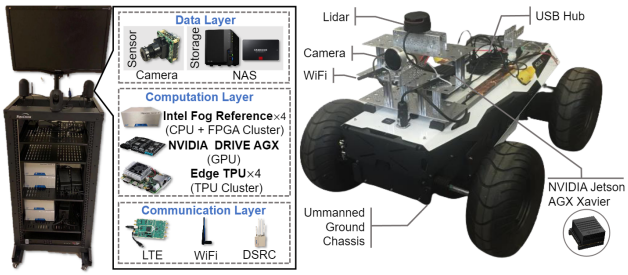

Fig. 4. The hardware overview of Equinox and Zebra.

1) Equinox: A Roadside Experimental Platform: Our EdgeServer, `Equinox`, is composed of three main layers: i) communication layer, ii) data layer, and iii) computation layer. The communication layer enables communication in three ways: WiFi, DSRC, and LTE. As to the data layer, a network-attached storage (NAS) and several Solid State Drives (SSD) are adopted. In addition, Equinox is designed with a heterogeneous computation layer, including four Intel Fog Reference Design-based CPU-FPGA, four Edge Tensor Process Units (TPUs), and an NVIDIA GPU. In particular, each Intel Fog Reference has a Xeon E3 CPU and Cyclone V FPGA board with 32GB memory. Each edge TPU board has an Integrated GC7000 GPU, 1GB RAM, and 8GB of flash memory. The NVIDIA DRIVE AGX Xavier GPU is the newest version of computing unit for autonomous driving vehicles and we use it to accelerate both model training and inference.

2) Zebra: A Robotic Vehicle Research Platform: Our designed robotic vehicle platform, `Zebra`, serves as the connected vehicle in the proposed VEC framework. It consists of three key components: sensors, computation units, and unmanned ground chassis (UGV). `Zebra` is equipped with two Intel Realsense L515 Lidar cameras[1] and one 2D Lidar (SLAMTEC RPLidar A3[2]) providing both color and depth information. The selected UGV is a Hunter 2.0 Mobile Robot of AgileX Robotics[3], which is designed as a programmable UGV for autonomous driving use cases. For communication, the HUNTER 2.0 provides Controller Area Network 2.0B (CAN2.0B) connectors [70] to establish communication with external devices. In particular, CAN2.0B is an extended version of the CAN protocol, supporting a communication baud rate of 500kbps using the Motorola message format. Regarding the computing unit, `Zebra` adopts the Nvidia Jetson AGX Xavier developer kit[4] for real-time, compute-intensive tasks, consuming around 10W of power. The LB-LINK Wi-Fi router supports 802.11a/b/g/n standards, enabling vehicles to connect with the RSU for measurement transmission.

D. Algorithm Description in VEC Framework

Next, we introduce and compare two RL policies within the VEC framework for adaptive compressive sensing. Furthermore, we evaluate the reconstruction quality of two models, E2E-CNN and BIRNAT, across various Cr values.

1) Adaptive Compressive Sensing: As described earlier (Sec.II-B), the key challenge for real-world CV applications is how to automatically and optimally adapt Cr under different driving scenarios. For example, in a slow-motion scenario (e.g., vehicles stopped at a red traffic light), a higher Cr is beneficial to reduce transmission bandwidth and accelerate inference speed while maintaining high detection accuracy. Conversely, in fast-motion scenarios (e.g., when vehicles accelerate), a lower Cr is required to prevent information loss.

In this work, we introduce RL [71] to the vehicle and combine RL with other machine learning methods (e.g., MOT models) to achieve adaptive temporal TCS. Briefly speaking, RL is a cyclical learning process in which an agent interacts with the unknown environment and takes action to change its state to maximize the expected cumulative reward. The goal of RL is to optimize behavioral strategies in sequential decision problems [72]. The RL agent will never be told the optimal action, instead, it will receive an evaluation signal (reward or penalty) indicating the goodness of the current action. Hence, RL matches well with an adaptive TCS considered in this work, where the vehicle usually does not know the environment and the objects in the scene being captured are dynamic and their speed can vary over time [73].

Within the VEC framework, the vehicle acts as the RL agent, taking responsibility for generating a specific reward (r) at each time step (t), i.e., $r_t \in R$. To enable CVs to automatically determine the optimal Cr, we assign a reward at each time step corresponding to the vehicle's forward action a = $\{decrease, keep, increase\}$, as predicted by the RL model. This includes adjusting Cr by either decreasing it, maintaining

[1]https://www.intelrealsense.com/lidar-camera-l515/
[2]https://www.slamtec.com/en/Lidar/A3
[3]https://global.agilex.ai/products/hunter-2-0
[4]https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit

its current value, or increasing it. For example, Cr can shift directly from 8 to 20 to better meet the dynamic demands of real-world applications. It is important to emphasize that this policy allows skipping intermediate Cr values. In this work, we posit that six reconstruction models are needed, each corresponding to different Cr values, i.e., Cr = {6, 8, 10, 12, 15, 20}, and have been trained for real-world applications. The selection of these Cr values is guided by heuristic methods, underpinned by extensive experimental analysis across various videos [55], to ensure satisfactory reconstruction quality. These values form a `state set` $\mathcal{S}$ = {6, 8, 10, 12, 15, 20}. S' represents the updated state after conducting each a. For each action step, the RL module issues a `reward` $r\,(S, a, S')$, correlating directly to the environmental context.

Table I. State transition table of the proposed RL for adaptive video TCS by only considering three states, i.e., $\mathcal{S}$ = {8, 12, 20}. Here, $increase^2$ denotes the jump from 8 to 20, and similarly, $decrease^2$ denotes the decrease from 20 to 8.

S	$action$	S'	p	r
8	decrease	8	0	–
8	keep	8	1	r_{keep}
8	increase	12	a	$r_{increase}$
8	$increase^2$	20	$1-a$	$r_{increase}$
12	decrease	8	1	$r_{decrease}$
12	keep	12	1	r_{keep}
12	increase	20	1	$r_{increase}$
20	$decrease^2$	8	b	$r_{decrease}$
20	decrease	12	$1-b$	$r_{decrease}$
20	keep	20	1	r_{keep}
20	increase	20	0	–

To be concrete, Table I presents the state transition through a simplified example, wherein Cr is limited to three distinct values: 8, 12, and 20, i.e., $\mathcal{S} = \{8, 12, 20\}$. In this scenario, initiating a search period with $S = 8$ does not permit a transition to a new state S' with the action $a = decrease$ since 8 represents the minimum value in the set; consequently, the related conditional probability $p = 0$, and no associated reward r is applicable. Conversely, with the action of increase, i.e., $\mathcal{S}$ = 8 and $a = increase$, $\mathcal{S}$ could be increased to 12 or 20 (i.e., S' = 12 or 20) with probability α and $1-\alpha$, respectively, where $\alpha \in [0, 1]$. Similarly, initiating a search with $\mathcal{S}$ = 20 and $a = decrease$ can result in a transition to $S' = 8$ with probability β and to $S' = 12$ with probability $1 - \beta$, where $\beta \in [0, 1]$.

2) Two Designed RL policies: In practical CV environments, the effectiveness of an RL policy is closely related to deep learning models and the characteristics of the driving scenarios. Initially, we consider the detection mAP (mean Average Precision) and the peak-signal-to-noise ratio (PSNR) of the reconstructed video as the key performance metrics for the RL module to adjust Cr. Here, PSNR is used to evaluate reconstruction model performance. However, considering real-world applications of our designed VEC framework (Fig. 3), we do not always have the detection rate due to the absence of ground truth in new driving scenarios. Also, PSNR is not consistently applicable, as video reconstruction is not always necessary, i.e., only when the number of detected vehicles exceeds a threshold (three), video reconstruction will be initiated, and the corresponding PSNR value will be obtained. Therefore, we propose two RL policies and evaluate them under highway and urban scenarios to facilitate adaptive TCS.

Specifically, we choose two objective metrics for adaptive Cr: i) the dynamic bounding box (bbox) size of the front vehicle tracked on captured measurements, and ii) the similarity of adjacent measurements. Given an initial Cr, RL will predict action a based on the analysis results (similarity or front bbox size) of captured measurements and keep the objective metric value within a suitable range. For instance, when the calculated similarity of adjacent measurements is smaller than the threshold, it indicates that the current Cr is larger than the optimal value, thus RL is expected to output a smaller Cr for the subsequent cycle.

To find the optimal range of Cr, we analyze the distribution of the objective metric values. This analysis enables us to identify the appropriate maximum and minimum thresholds for Cr: upper/lower bound = $min + (max - min) \times \lambda$ ($\lambda_1 \in (0, 1)$), where the min and max refer to the minimum and maximum value, and λ is a coefficient to distinguish the upper and lower bound threshold. As to the front vehicle's bbox size, we set $\lambda = 0.7$ and $\lambda = 0.4$ to get the upper bound (23,400 pixels) and lower bound threshold (16,398 pixels), respectively. Without loss of generality, we set $\lambda = 0.94$ and $\lambda = 0.9$ for the similarity metric to get the upper bound (0.9417) and lower bound threshold (0.9333).

(1) RL-I: Tracking-Driven Adaptive TCS. This policy leverages MOT algorithms for adaptive TCS in vehicle applications. Specifically, we use MOT models to track the front vehicle and obtain its bbox size in pixels, with RL-I aiming to maximize cumulative reward based on real-time bbox size. Then, we train, deploy, and compare two SOTA tracking models on our proposed platform: (1) Deep SORT [40], which employs a tracking-after-detection approach, and (2) CenterTrack [41], which integrates detection and tracking within a unified framework. These models are extended to incorporate TCS by dynamically adjusting Cr based on the tracked front vehicle's bbox size. Both models utilize different distance metrics for object association: cosine distance compares object appearance across frames [74], while Mahalanobis distance considers object location and scale, enhancing robustness in complex driving scenarios [75]. These metrics play a key role in the effectiveness of tracking algorithms in dynamic environments [40], [41].

Deep SORT: Deep SORT [40] operates in two phases: (i) the detecting phase, and (ii) the tracking phase. In the detection phase, targets are identified, and bounding boxes are computed. For each bounding box, Deep SORT generates a 128-dimensional appearance descriptor and motion descriptor. Then, it combines the cosine distances of the appearance descriptors with the Mahalanobis distances of the motion descriptors to determine the admissibility of frame-by-frame associations, enabling effective tracking of multiple targets.

CenterTrack: Different from the "tracking after detection"

paradigm (e.g., Deep SORT), CenterTrack [41] is a typical example of "joint detection and tracking" and "tracking objects as points". Firstly, CenterTrack simplifies tracking-conditioned detection. It jointly infers all objects when associating them across frames. Secondly, CenterTrack tracks objects based on points which can simplify target association over time, as it can jointly detect points for the current frame and associate the current target with prior detection results.

In this context, the RL-1 strategy optimizes cumulative rewards by leveraging real-time data on the bbox size of the front-most vehicle. The rationale for relying on the bbox size to guide RL decisions (e.g., adjusting Cr) is two-fold:

- The change in the front vehicle's bbox size can serve as an indicator of both the host vehicle's speed and that of the front vehicle. For instance, if the bbox of the front vehicle enlarges, it could suggest that the host vehicle has entered a slow-motion scenario. Under such circumstances, Cr can be increased to facilitate faster processing speeds while still retaining essential information.
- The change in the front vehicle's bbox size also provides insights into the distance between it and the host vehicle. A smaller bbox may imply a greater distance, with each object occupying fewer pixels in the image. Hence, the Cr should be reduced in such scenarios; otherwise, a larger Cr could exacerbate the challenge of extracting information from the corresponding measurement.

We specifically focus on the bbox size of the front vehicles, designated with a vehicle ID of 1, while excluding other detected vehicles from consideration. The front vehicle's ID, as determined by the MOT algorithm, is less prone to reassignment or disappearance during the tracking phase, thereby ensuring greater reliability in the tracking outcomes.

(2) RL-II: Similarity-Driven Adaptive TCS. The second proposed RL policy focuses on analyzing the similarity between consecutive measurements. By employing MobileNets [76] for feature extraction from the measurements, we compute the cosine distance between the features of adjacent measurements, yielding similarity scores within the range of [-1, 1]. A score of 1 indicates maximum similarity, whereas -1 signifies the utmost dissimilarity. The reasons for utilizing measurement similarity to guide RL decisions include:

- A substantial change in similarity can indicate alterations in velocity or transitions between different driving scenes. For example, a decrease in similarity, which indicates a larger gap between consecutive measurements, may imply rapid scene changes (e.g., a vehicle navigating into a new block), necessitating a reduction in Cr to accommodate the increased information variability.
- Variations in similarity also reflect the speed of the host vehicle. As the vehicle accelerates, the similarity between successive measurements decreases, indicating a fast-motion scenario. This suggests that reducing Cr is essential to accurately capture transient high-speed data without compromising information integrity.

E. Video Reconstruction

We introduce E2E-CNN and BIRNAT for video reconstruction, both trained on measurements with different Cr values. We then compare the reconstruction quality of adaptive Cr against non-adaptive compression with fixed Cr, as discussed in Sec. IV-B, Sec. IV-B5, and Sec. IV-C.

E2E-CNN [28] is proposed to enable a millisecond-level reconstruction for TCS problems. Specifically, E2E-CNN includes a convolutional encoder and decoder with residual block connection (res-block) [14], where both the encoder and decoder consist of five residual blocks, and they are connected by two convolutional layers. Different from the conventional iteration-based methods [77], which require iteration and computation for each measurement, E2E-CNN conducts optimization only during training and efficiently recovers images during inference.

BIRNAT [29] is proposed to reconstruct the first frame as a reference for the reconstruction of the following frames through a measurement pre-processing method. The first (reference) frame is reconstructed by an attentional res-block based convolutional neural network (CNN), and the following frames are sequentially inferred by a bidirectional recurrent neural network (RNN) based on the first frame. The experiment results on both simulation and real datasets demonstrate that BIRNAT outperforms current SOTA algorithms, including GAP-TV [78], DeSCI [77], and U-net [28].

IV. Evaluation Result and Discussion

In this section, we meticulously present and analyze the comprehensive evaluation results of the proposed VEC framework, providing valuable insights into its performance and effectiveness. Specifically, we assess the proposed framework in the following ways: i) measurement-based object detection accuracy and reconstruction quality of various algorithms under diverse Cr, ii) performance of two reinforcement learning policies for adaptive TCS, and iii) computational latency, bandwidth reduction, and resource utilization, including CPU, GPU, and memory footprint.

A. Measurement-based Vehicle Detection under Adverse Environments

As discussed in Section III-A, it is challenging to guarantee the performance of models in inclement weather such as rain and snow, where data variation is unpredictable and difficult to capture or utilize [79]. Additionally, inference accuracy can significantly decrease from approximately 96% to 36% when the execution environment shifts from normal to adverse scenarios [80], [81]. This decline occurs because many models are trained on standard and clean datasets like COCO [82], causing them to fail in accurately recognizing objects in adverse conditions. Motivated by this insight, we pre-trained an advanced object detection model on the cloud using data specifically collected under normal, rainy, and snowy conditions for vehicle model updating purposes. The cloud also hosts an increasing amount of global data with diverse

geographic, environmental, and weather conditions, which is beneficial for continuously training more advanced models.

To prove the effectiveness of measurement-based detection under adverse environments, we trained the YOLOv3-Tiny model [83] on three groups of measurements under normal, rainy, and snowy conditions. Fig. 5 presents an example of detection results. In rainy and snowy days, the sliding of the vehicle windshield wipers causes obvious noise on the measurement (as shown in Fig. 5(b) and Fig. 5(c)), but this does not affect the object detection on the measurement. We employ a YOLOv3 network on the original public video dataset to get labels (bounding boxes of vehicles) and treat these labels as the ground truth (pseudo-label). We use mAP (mean Average Precision) as our detection rate score [84]. Specifically, as we focus on a single category, mAP corresponds to Average Precision (AP), which represents the area under the precision-recall curve, providing a summary of the model's performance across varying thresholds. We also consider the intersection-over-union (IoU) [85] between ground truth and predictions as an additional evaluation metric.

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} = \frac{\text{TP}}{\text{all detections}} \tag{1}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} = \frac{\text{TP}}{\text{all ground truths}} \tag{2}$$

$$\text{IoU} = \frac{\text{area}\left(bbox_p \cap bbox_{gt}\right)}{\text{area}\left(bbox_p \cup bbox_{gt}\right)} \tag{3}$$

where TP is the number of detection frames with IoU (intersection-over-union) > 0.5 and FP with IoU $\leqslant 0.5$ detection frames. FN refers to the number of missing detections. $bbox_{gt}$ represents the bbox of ground truth (GT) and $bbox_p$ of the predicted frame.

(a) Normal day (b) Rainy day (c) Snowy day

Fig. 5. An example of vehicle detection on three groups of measurements under different weather conditions.

Figure 5 presents an example of detection results. In rainy and snowy days, the sliding of the vehicle windshield wipers causes obvious noise on the measurement (as shown in Fig. 5(b) and Fig. 5(c)), but this does not affect the object detection on the measurement.

B. Results of RL-I: Tracking-Driven Adaptive TCS

In this work, we evaluate the reconstruction quality of various algorithms under diverse Cr values, including two RL policies, separately.

1) Evaluation Metric: To explore the suitable tracking algorithm, we train and test Deep SORT and CenterTrack on measurements. Then, we use the MOTA to describe model performance on multi-object tracking tasks, defined as follows:

$$\text{MOTA} = 1 - \frac{\Sigma_t(FP_t + FN_t + IDSW_t)}{\Sigma_t CT_t} \tag{4}$$

Where FP_t, FN_t, $IDSW_t$, and CT_t represent false positives, false negatives, identity switches, and the ground truth in measurement frame t.

Additionally, to assess the quality of the reconstructed images, we employ the Peak Signal-to-Noise Ratio (PSNR) as the evaluation metric. PSNR [86] measures the ratio between the peak signal and the noise level in the reconstruction relative to the original image. By comparing PSNR values, we can clearly understand the performance of the reconstruction images. More specifically, given an original image I of size $m \times n$ and an image K after reconstructing, the mean squared error (MSE) is defined as follows, and the average PSNR of the video group is given by:

$$\text{PSNR} = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) \tag{5}$$

$$\text{MSE} = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i-j) - K(i-j)]^2 \tag{6}$$

Where MAX_I^2 denotes the maximum possible pixel value in the image I, i and j are indices that are used to iterate over the pixels. This formulation allows for the adaptation of the PSNR calculation to various image formats by accurately defining the maximum pixel intensity based on the bit depth.

2) Deep SORT vs. CenterTrack: In our test, CenterTracker showed better tracking accuracy: with a multi-object tracking accuracy (MOTA) value of 74.42, higher than DeepSort (60.85). Besides, the average inference time of CenterTrack is only around one-third of Deep SORT's average inference time (i.e., 26ms vs. 86ms). Therefore, we adopt CenterTrack to support the adaptive TCS based on vehicle tracking.

3) The Effectiveness of RL-I: Fig. 6 presents the adaptive TCS results based on the bbox size of the front vehicle that is tracked from the measurements under highway scenarios, to detect surrounding vehicles from the *adaptive measurements*.

Specifically, Fig. 6(a) presents the changes in the front vehicle's bbox size (in pixels) and *adaptive* Cr (frames) from the measurements against a constant stream of measurements. As shown in Fig. 6(a), Cr has approximately maintained a certain range at the beginning. Then the host vehicle observes a bridge ahead and slows down its speed. This change leads to a larger front vehicle's bbox, consequently, Cr rises to a higher level (Cr = 8) and finally rises to the defined highest value (Cr = 20). In the end, Cr drops back to the original lower level (Cr = 6) when the vehicle speed becomes normal to pass through the bridge.

Three normalized measurements with different values of adaptive Cr are shown in Fig. 6 (b-d) with adaptive Cr = 6, 20, 6. We can see that the normalized measurement (c) has the largest adaptive Cr = 20 since its corresponding original video frames are moving slowly, while the normalized measurement (d) is blurry with the smallest adaptive Cr = 6 due to the fast vehicle speed in associated video frames. Accordingly, the RL-I enables dynamic adjustment of Cr based on the front vehicle's bbox size, ensuring optimal adaptation to varying vehicle speeds.

4) Adaptive TCS vs. Non-adaptive TCS: In Fig. 6, the video has a total number of 1494 frames, achieving an average compression ratio (average Cr) of 10.89. To demonstrate the

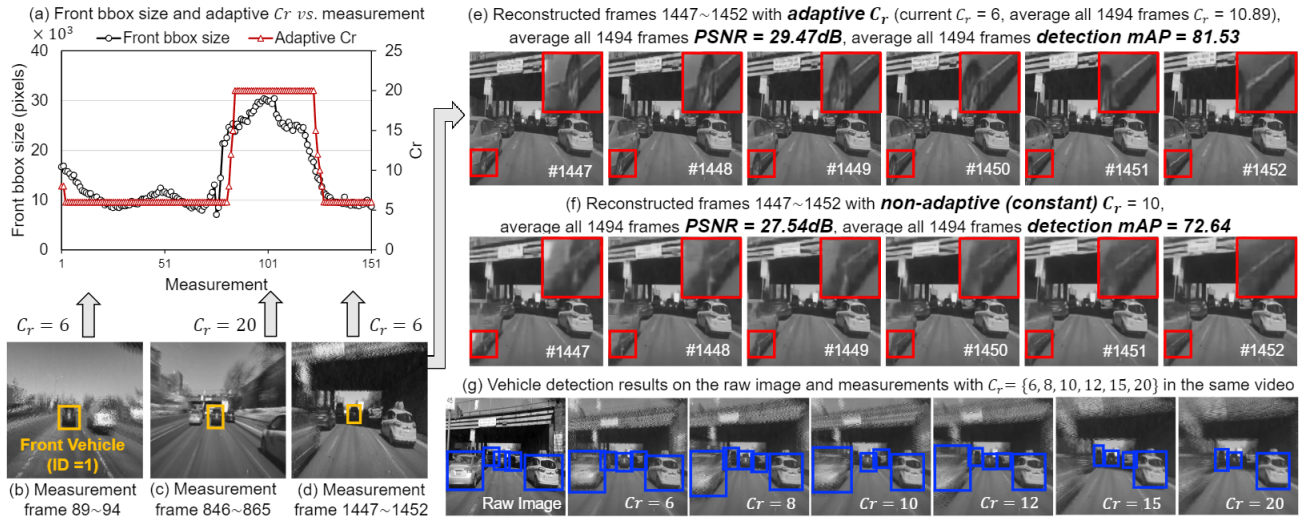

Fig. 6. Adaptive Cr based on the bbox size of front vehicle that is *tracked from the measurements directly*. (a) Reconstruction PSNR (dB) and adaptive Cr (frames) (average adaptive Cr = 10.89) from the *measurements*, all are plotted against the index number of measurements. (b-d) Normalized measurements when the vehicle passes through the bridge, adapted Cr = 6, 20, 6, respectively. (e) Reconstructed frames 1447∼1452 from the measurement in (d) with *adaptive* Cr. (f) Reconstructed frames 1147∼1452 with non-adaptive (constant) Cr = 10. (g) Vehicle detection results on the raw images and measurements with different $Cr = \{6, 8, 10, 12, 15, 20\}$ in the same video clip.

advantages of adapting Cr, we compare adaptive reconstructions (Fig. 6(e)) to those obtained when Cr is fixed at or near its expected value (Fig. 6(f)) at Cr = 10. Fig. 6(f) shows the reconstructed frames 1147∼1452 from the measurement in (d) with *non-adaptive* (constant) Cr. Comparing Fig. 6(e) and 6(f), we notice that adapting Cr provides a higher (1.93dB) reconstruction quality (average all 1494 frames PSNR=29.47dB) than fixing Cr even lower than its expected value (average PSNR = 27.54dB). Besides, it also improves the average detection mAP from 72.64 to 81.53. To present the effects of diverse Cr on object detection based on measurements, we visualize the vehicle detection results on the raw (original) images and measurements with different Cr = $\{6, 8, 10, 12, 15, 20\}$ in the same video clip in Fig. 6(g). It can be seen that a decent detection mAP is obtained at Cr = 6 or 8, while a larger Cr will lead to false alarms. Consequently, compared to the reconstructed video, both mAP and PSNR are higher (mAP by 8.89, PSNR at 29.5).

5) Video Reconstruction: To figure out a suitable reconstruction method, we train and compare two SOTA algorithms, i.e., E2E-CNN and BIRNAT with different Cr.

Table II. Reconstruction Quality (PSNR in dB) Comparison.

Algorithm	Cr = 6	Cr = 8	Cr = 10	Cr = 12	Cr = 15	Cr = 20
E2E-CNN	27.76	26.40	25.77	24.74	24.52	23.33
BIRNAT	34.71	32.60	29.97	29.41	29.32	28.48

Table II presents the reconstruction results (i.e., the change in the value of PSNR) with different Cr. A higher PSNR value indicates a better reconstruction quality. It can be seen that BIRNAT achieves higher PSNR values than E2E-CNN across all Cr. Therefore, we deploy BIRNAT at EdgeServer for video image reconstruction (Fig. 3).

C. Results of RL-II: Similarity-Driven Adaptive TCS

Similar to Fig. 6, Fig. 7 shows how the adaptive Cr changes based on the similarity of measurements in urban areas. Here, the similarity of adjacent measurements has more obvious fluctuations since the background information is richer than the highway.

1) The Effectiveness of RL-II: As shown in Fig. 7(a), in the middle part, the similarity has a relatively sharp drop, and this is because the host vehicle turns left into another road, i.e., there is a dramatic transition from one driving scene to another. In this context, Cr should be reduced in theory. However, since the associated Cr achieves the defined lowest value (Cr = 6), RL decides to keep the lowest Cr until the host vehicle completely drives into the new district.

2) Adaptive TCS vs. Non-adaptive TCS: From the reconstructed frames in Fig. 7(e)-(f) and detection frames in (g), we can see that adapting Cr leads to a 2.31dB improvement in PSNR and an increase of 5.69 in vehicle detection mAP, illustrating the effectiveness of adaptive TCS. Overall, the RL-II dynamic adjustment of Cr achieves higher mAP (68.83 vs. 74.97) and PSNR (25.97 vs. 28.28).

D. Inference Latency of Involved Models

The VEC framework involves four types of models: i) object detection (including YOLOv3-Tiny and YOLOv3), ii) reconstruction (E2E-CNN and BIRNAT), iii) MOT (CenterTrack and Deep SORT), and iv) Similarity analysis, i.e., MobileNets. A prior study reports that the execution time for each real-time task pipeline should remain below 100ms for an autonomous vehicle traveling at 40km/h in urban environments [1]. As presented in Table III, we calculate the average latency of our task pipeline to demonstrate its efficiency and practicality in real-world CV applications. The methodology for calculating total latency is detailed in Sec. IV-G2.

Table III. Average inference time of deep learning models.

Model	YOLOv3-Tiny	YOLOv3	E2E-CNN	BIRNAT	CenterTrack	Deep SORT	MobileNets
Time(ms)	16	42	29	96	26	86	39

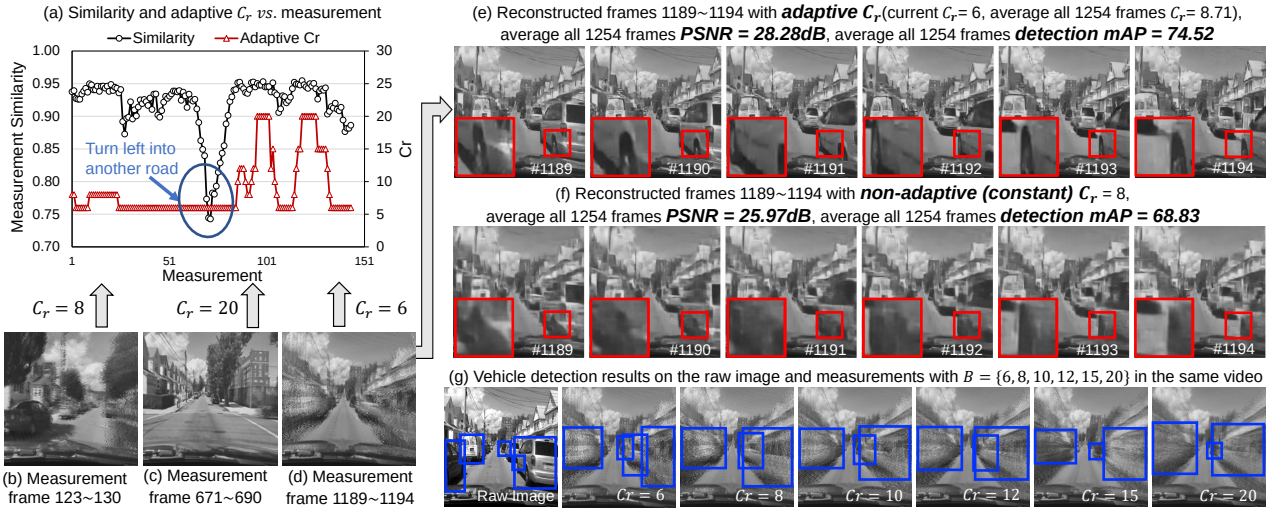

Fig. 7. Adaptive Cr based on *similarity of adjacent measurements*. (a) Reconstruction PSNR (dB) and adaptive Cr (frames) (average adaptive Cr = 8.71) from the *measurements*, all are plotted against measurement numbers. (b-d) Normalized measurements when the vehicle drives in the urban district, adapted Cr = 8, 20, 6, respectively. (e) Reconstructed frames 1189∼1194 from the measurement in (d) with *adaptive* Cr. (f) Reconstructed frames 1189∼1194 with non-adaptive (constant) Cr = 8. (g) Vehicle detection results on the raw images and measurements with different $Cr = \{6, 8, 10, 12, 15, 20\}$ in the same video clip.

E. Transmission Bandwidth

1) Correctness Checking for Video Transmission: The Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP) are the dominant communication protocols [87]. In this work, we use UDP for measurement transmission. Since there is no sequencing of data in UDP, in the process of data transmission and receiving, data disorder will occur. To overcome this issue, we conduct correctness checking by adding source and destination IP addresses to the header along with parity checking.

2) Theoretical Value: In this work, only (compressed) measurements will be transmitted. Theoretically, with a defined Cr, the transmission bandwidth (considering a single channel) can be decreased to $\frac{1}{3\times Cr}$ of what would be required for transmitting raw RGB images (which involve three channels). This efficiency enables the feasibility of our framework on any communication platform that offers bandwidth exceeding 320KB/s, as detailed in Table IV.

Table IV. Real-time transmission bandwidth requirements

Image Type	RGB	Grey-scale	Measurement (Cr = 6)	Measurement (Cr = 15)	Measurement (Cr = 20)
EdgeServer Recv	30fps	30fps	5fps	2fps	1.5fps
Available Time	33.33ms	33.33ms	200ms	500ms	666.67ms
Bytes Per Frame	196641	65569	65569	65569	65569
Required Bandwidth	5760.97KB/s	1920.97KB/s	**320.16KB/s**	128.06KB/s	96.05KB/s

We first focus on the ideal case (i.e., only consider real-time image transmission). Suppose the original video stream on the vehicle is 30fps, then to achieve real-time transmission, EdgeServer should process no less than 30 measurements per second. Table IV presents the theoretical values of five metrics to achieve real-time, including

- `EdgeServer Recv`: the minimum number of frames received by EdgeServer per second,
- `Available Time`: the maximum delay allowed for each frame to be analyzed and transmitted,
- `Bytes Per Frame`: the number of bytes contained in each frame (an image is converted into a byte stream),
- `Required Bandwidth`: the minimum bandwidth required.

As shown in Table IV, we summarize the transmission requirements for each image category: i) raw RGB images, ii) grey-scale images, iii) measurements with Cr = 6, iv) measurements with Cr = 15, and v) measurements with Cr = 20, to achieve real-time transmission. Recall that we assume the original video stream on the vehicle is 30fps, and thus in this context, `EdgeServer Recv` of RGB and grey-scale images should achieve at least 30fps for real-time applications. However, as the "measurement with Cr = 6" group and the "measurement with Cr = 15" group, they only receive 5 and 2 frames per second, respectively. In this context, `EdgeServer Recv` of RGB and grey-scale images are both equal to around 33.33ms ($\frac{1s}{30fps}$); however, as to the two measurement groups (i.e., Cr = 6 and Cr = 8), the minimum number of frames received per second can increase to 200ms ($\frac{1s}{5fps}$) and 500ms ($\frac{1s}{2fps}$). This means that conducting analysis on measurements rather than RGB images could significantly increase (greater than 6×) the maximum delay allowed to achieve real-time transmission. In addition, without TCS, the average inference time of many models, such as YOLOv3, MobileNets, and Deep SORT (Table III), will exceed 33.33ms.

As to the `Bytes Per Frame`, the number of bytes in RGB frame is around 3× that grey-scale images and measurements (all images and measurements are in the size of 256 × 256 pixels). Based on the definition, `Required Bandwidth` = $\frac{\texttt{Bytes Per Frame}\times\texttt{EdgeServer Recv}}{1s}$, we can calculate the value of `Required Bandwidth` (shown in Table IV) for real-time applications. Recall that our measured

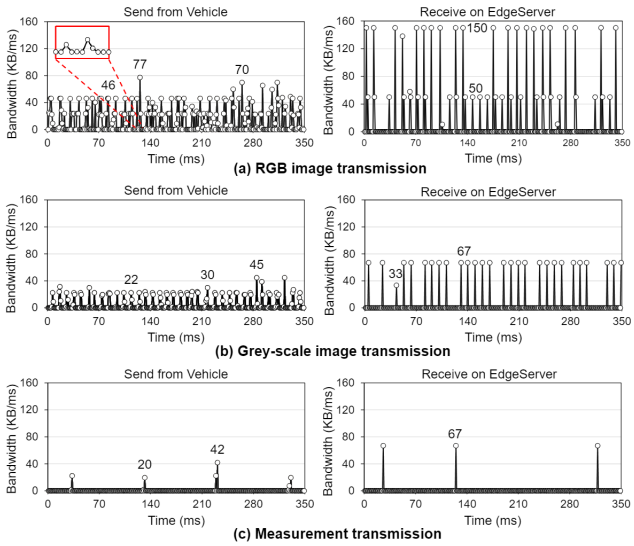

Fig. 8. An example of *communication bandwidth* evaluation results between vehicle and EdgeServer.

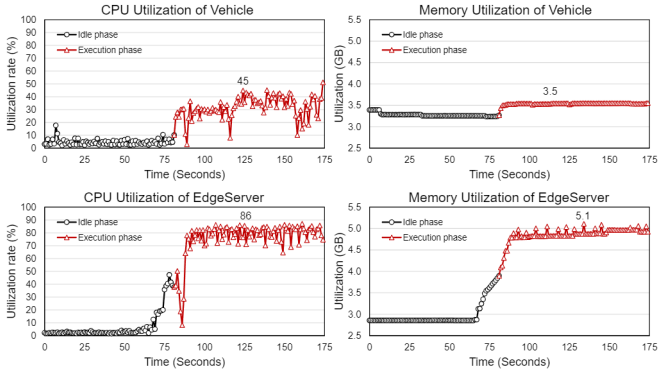

Fig. 9. An example of communication bandwidth evaluation results between vehicle and EdgeServer.

bandwidth is around 4000KB/s, which is much smaller than RGB's `Required Bandwidth` (5760.97KB/s). This highlights a critical finding: real-time applications would be infeasible without the use of TCS. Most importantly, the `Required Bandwidth` of RGB images is around $18\times$ that of measurements with Cr = 6, and this difference grows exponentially when continuously increasing Cr. This observation indicates the feasibility and promise of realizing real-time CV applications. Further observations and discussions are provided in Sec. V.

3) Experiment Value: To quantify the real performance of bandwidth reduction, we monitor and record the transmission bandwidth from the perspective of the vehicle and the EdgeServer respectively, specifically the sending process from the vehicle and the receiving process on the EdgeServer. Figure 8 presents the bandwidth utilization (KB/ms) of these two processes for three groups, i.e., RGB image transmission (Fig. 8(a)), grey-scale image transmission (Fig. 8(b)), and measurement transmission (Fig. 8(b)) with adaptive Cr. It is clear RGB image transmission leads to the highest bandwidth pressure on the network than gray-scale image transmission, and the measurement transmission has the lowest bandwidth requirements with the lowest frequency of bandwidth peak. Besides, given a specific group, the value of bandwidth peaks in the sending process is significantly less than that of the receiving process; however, the frequency of bandwidth peaks in sending is greater than that of the receiving process.

F. CPU and Memory Utilization

We then evaluate the CPU and memory utilization on both the vehicle and the EdgeServer. As shown in Fig. 9, the black color represents the idle phase, and the red color highlights the execution phase. It can be observed that the EdgeServer has a higher CPU utilization rate and memory footprint. This observation also indicates that our proposed VEC framework can transfer a significant portion of the CPU and memory utilization pressure from the vehicle to the EdgeServer.

G. Additional Considerations

1) Robustness to Noise: We also assess the proposed RL module's robustness to noise by evaluating its ability to recover from noisy measurements. As indicated in Table V, even after adding zero-mean Gaussian noise $\boldsymbol{n} \sim \mathcal{N}(0, \sigma)$ to measurements normalized within the range of $[0, 1]$, the reconstruction quality (reflected by PSNR in dB) and detection mAP remain high.

2) Inference Speed: A prior study indicated that for an autonomous vehicle traveling at 40km/h in urban areas, the execution time for each real-time task pipeline should not exceed 100ms [1]. In this work, two key applications are implemented on the vehicle: object detection and RL policies, which encompass MOT and similarity analysis. Only one RL policy is executed on the vehicle, resulting in a total pipeline latency of 42ms, comprising 16ms for YOLOv3-Tiny and 26ms for CenterTrack. Reconstruction is performed on the RSU rather than on the vehicle, as the detection accuracy on compressed measurements is comparable to that on reconstructed videos, with evaluation metrics closely aligning with true values [55]. Thus, the total inference time on the vehicle remains well below the 100ms threshold, demonstrating the practicality and efficiency of employing our framework for real-world CV applications.

Table V. Reconstruction PSNR and detection mAP *vs.* noise σ.

PSNR, DR / B / σ	**6**	**10**	**15**
0	28.73, 85.43	28.44, 85.57	28.33, 81.38
0.005	28.56, 85.21	28.30, 84.36	28.19, 80.18
0.010	28.18, 83.74	27.99, 81.62	27.89, 77.45
0.050	24.70, 75.34	24.62, 76.33	24.52, 71.26
0.100	21.58, 71.47	21.52, 71.23	21.44, 68.49

3) Practicality to Real Systems: Moreover, recent advances in reconstruction networks have demonstrated excellent performance through offline training on simulated data [88]. We believe that training the RL model on simulated data and applying it to real data will yield similar results. The robotic vehicle is equipped with industry-grade computing boards and sensors, closely replicating a real-world vehicle testbed. As a result, the experimental outcomes from these testbeds offer practical insights that are directly transferable to real vehicle applications in actual driving scenarios.

V. Observation and Discussion

In this section, we present our main experiment findings and discuss the key observations.

⋆ **Observation 1:** With the integration of TCS, our proposed framework not only maintains high application accuracy on compressed measurements but also reduces vehicle transmission bandwidth from 5761KB/s to 320KB/s, achieving an over $18\times$ reduction compared to existing systems.

The effectiveness of the TCS-driven video process is evidenced by findings in Sec. IV-A, Sec. IV-D, and Sec. IV-E.

Discussion: Nowadays each vehicle typically generates approximately 8GB of data per day whether the vehicle is stationary or in motion. Although 5G technology improves spectrum and energy efficiency, the EdgeServer still encounters challenges related to traffic volume and network quality of service. One effective way to mitigate bandwidth overhead is by reducing the amount of data transmitted from each vehicle. In this work, if 200 raw data frames per second need to be transmitted, adjusting the Cr based on different driving scenarios to implement TCS is able to reduce this burden. For example, when $Cr = 10$, we collect only 20 measurements per second, replacing the 200 raw frames. Consequently, our framework transmits only the measurements, achieving more than an $18\times$ reduction in both communication bandwidth and latency. Furthermore, increasing Cr to a larger value, such as $Cr = 20$, would yield even greater reductions in bandwidth usage and latency compared to $Cr = 10$.

⋆ **Observation 2:** Our RL agent effectively demonstrates its ability to dynamically determine the optimal Cr based on varying driving scenarios, resulting in improved PSNR and mAP compared to non-adaptive TCS.

This observation is supported by the comparison results between Fig. 7(e) and Fig. 7(f) as well as frames with detection results in Fig. 7(g).

Discussion: The adaptive TCS demonstrates superior performance over the non-adaptive approach, resulting in higher-quality compressed image data, which is essential for maintaining accuracy in real-time vehicle applications (e.g., object detection). Specifically, the adaptive Cr yields higher PSNR by 2.31dB and mAP by 5.69 for vehicle detection compared to the non-adaptive Cr. Moreover, adaptive Cr minimizes the risk of information loss in fast-motion scenarios while reducing unnecessary bandwidth consumption in slow-motion situations, thus achieving both bandwidth efficiency and high detection accuracy. These illustrate our RL agent's ability to adapt Cr based on real-time feedback allowing for more accurate object detection. Due to the high detection accuracy of compressed image data, the onboard computation can be further reduced by offloading the resource-intensive reconstruction process from the vehicle to the EdgeServer.

⋆ **Observation 3:** In terms of network strain, measurement data transmission imposes the least bandwidth, followed by grayscale image transmission, with raw RGB image transmission causing the highest strain. Additionally, the sending process exhibits a higher frequency of bandwidth peaks compared to the receiving process.

This observation is supported by Fig. 8(a), Fig. 8(b), and Fig. 8(c).

Explanation and Discussion: Compared to gray-scale image transmission, measurement transmission typically consists of compact numerical values or metadata, resulting in minimal bandwidth usage. In contrast, raw RGB image transmission imposes the greatest strain on the network due to its three-channel structure, which transmits large volumes of uncompressed pixel-level data. Also, this high data load significantly increases bandwidth consumption and leads to the rapid transmission of large data volumes, causing bandwidth peaks.

⋆ **Observation 4:** The EdgeServer exhibits higher CPU utilization and a larger memory footprint, indicating that our proposed VEC framework efficiently transfers a considerable portion of the vehicle's computational and memory-intensive tasks to the EdgeServer.

This observation is evidenced by Fig. 9.

Discussion: Our designed VEC framework is designed to distribute data processing and computational tasks between the vehicle and the EdgeServer. By offloading resource-intensive tasks, such as image decoding, data compression, and real-time analysis, to the EdgeServer, the vehicle's onboard system experiences reduced strain on its CPU and memory. This shift allows the vehicle to focus on critical real-time operations like navigation or object detection, while the EdgeServer handles more demanding computational workloads. These highlight the efficiency of the VEC framework in optimizing the vehicle's performance by leveraging external computing resources.

VI. Conclusion and Future Work

In this work, we propose a VEC framework and driving-aware compression mechanisms to manage continuous big data transmission in CVs. By employing two RL policies and TCS, we reduce bandwidth requirements by up to $18\times$ at 320KB/s while retaining critical information for real-time applications (e.g., object detection). Our lightweight vehicle model's real-time detection capability, coupled with EdgeServer reconstruction when necessary, demonstrates a robust approach to balancing resource limitations and maintaining high detection accuracy and reconstruction quality, compared to non-adaptive measurements. This highlights the framework's promising real-world applications for CVs. In the future, we will focus on enhancing the framework to support seamless scalability with multiple vehicles, enabling efficient data transmission, communication, and coordination across large fleets without compromising performance.

Acknowledgement

This work is supported in part by the National Science Foundation (NSF) grant CNS-2348151, CNS-2231523, and Commonwealth Cyber Initiative grant HC-3Q24-048.

REFERENCES

[1] S. Lu and W. Shi, "Vehicle as a mobile computing platform: opportunities and challenges," *IEEE Network*, 2023.

[2] F. Ahmad, H. Qiu, R. Eells, F. Bai, and R. Govindan, "CarMap: Fast 3D feature map updates for automobiles," in *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2020, pp. 1063–1081.

[3] X. Yang, X. Wang, Z. Li, Y. Liu, F. Qian, L. Gong, R. Miao, and T. Xu, "Fast and light bandwidth testing for internet users." in *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2021, pp. 1011–1026.

[4] T. Giuffrida, "Enabling the connected vehicle market to thrive," Available: https://aecc.org/resources/publications/, March 2020, accessed: Mar. 2020.

[5] N. Varga, L. Bokor, A. Takács, J. Kovács, and L. Virág, "An architecture proposal for v2x communication-centric traffic light controller systems," in *2017 15th International Conference on ITS Telecommunications (ITST)*. IEEE, 2017, pp. 1–7.

[6] Microsoft News Center, "Cruise and GM team up with Microsoft to commercialize self-driving vehicles," Available: https://news.microsoft.com/2021/01/19/cruise-and-gm-team-up-with-microsoft-to-commercialize-self-driving-vehicles/, January 2021, accessed: Jan. 19, 2021.

[7] A. Sam, "Automotive Over-The-Air Updates: A Cost Consideration Guide," Available: https://www.auroralabs.com/wp-content/uploads/2021/05/OTA_Update_Cost_Consideration_Guide_Apr2021.pdf, April 2021, accessed: Apr. 2021.

[8] T. Lida, "Is data transmission the new fuel?" Available: https://www.auroralabs.com/is-data-transmission-the-new-fuel/, June 2021, accessed: June. 15, 2021.

[9] S. Kato, E. Takeuchi, Y. Ishiguro, Y. Ninomiya, K. Takeda, and T. Hamada, "An open approach to autonomous vehicles," *IEEE Micro*, vol. 35, no. 6, pp. 60–68, 2015.

[10] B. Yu, W. Hu, L. Xu, J. Tang, S. Liu, and Y. Zhu, "Building the computing system for autonomous micromobility vehicles: Design constraints and architectural optimizations," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2020, pp. 1067–1081.

[11] J. Chen and S. Lu, "An advanced driving agent with the multimodal large language model for autonomous vehicles," in *2024 IEEE International Conference on Mobility, Operations, Services and Technologies (MOST)*. IEEE, 2024, pp. 1–11.

[12] Y. Luo, D. Xu, G. Zhou, Y. Sun, and S. Lu, "Impact of raindrops on camera-based detection in software-defined vehicles," in *2024 IEEE International Conference on Mobility, Operations, Services and Technologies (MOST)*. IEEE, 2024, pp. 193–205.

[13] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2015, pp. 1–9.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2016, pp. 770–778.

[15] A. Chehri, H. Chehri, N. Hakim, and R. Saadane, "Realistic 5.9 GHz DSRC vehicle-to-vehicle wireless communication protocols for cooperative collision warning in underground mining," in *Smart Transportation Systems 2020*. Springer, 2020, pp. 133–141.

[16] S. A. AbdelHakeem, A. A. Hady, and H. Kim, "Optimizing 5G in V2X communications: Technologies, requirements, challenges, and standards," in *Research Anthology on Developing and Optimizing 5G Networks and the Impact on Society*. IGI Global, 2021, pp. 972–1011.

[17] Q. Luo, C. Li, T. H. Luan, W. Shi, and W. Wu, "Self-learning based computation offloading for internet of vehicles: Model and algorithm," *IEEE Transactions on Wireless Communications*, 2021.

[18] M. Qiao, X. Liu, and X. Yuan, "Snapshot spatial–temporal compressive imaging," *Opt. Lett.*, vol. 45, no. 7, pp. 1659–1662, Apr 2020.

[19] L. Cheng, Z. Wu, R. Duan, and K. Dong, "Adaptive compressive sensing and machine learning for power system fault classification," in *2020 SoutheastCon*. IEEE, 2020, pp. 1–7.

[20] K. Sever, T. Vlašič, and D. Seršič, "A realization of adaptive compressive sensing system," in *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO)*. IEEE, 2020, pp. 152–157.

[21] M. F. Duarte, M. A. Davenport, D. Takhar, J. N. Laska, T. Sun, K. F. Kelly, and R. G. Baraniuk, "Single-pixel imaging via compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 83–91, 2008.

[22] Z. Wang, H. Zhang, Z. Cheng, B. Chen, and X. Yuan, "MetaSCI: Scalable and Adaptive Reconstruction for Video Compressive Sensing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 2083–2092.

[23] X. Yuan, J. Yang, P. Llull, X. Liao, G. Sapiro, D. J. Brady, and L. Carin, "Adaptive temporal compressive sensing for video," in *2013 IEEE International Conference on Image Processing (ICIP)*, Sept 2013, pp. 14–18.

[24] S. Jalali and X. Yuan, "Snapshot compressed sensing: Performance bounds and algorithms," *IEEE Transactions on Information Theory*, vol. 65, no. 12, pp. 8005–8024, Dec 2019.

[25] M. Iliadis, L. Spinoulas, and A. K. Katsaggelos, "Deep fully-connected networks for video compressive sensing," *Digital Signal Processing*, vol. 72, pp. 9–18, 2018.

[26] J. Ma, X. Liu, Z. Shou, and X. Yuan, "Deep tensor admm-net for snapshot compressive imaging," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[27] K. Xu and F. Ren, "CSVideoNet: A real-time end-to-end learning framework for high-frame-rate video compressive sensing," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2018, pp. 1680–1688.

[28] M. Qiao, Z. Meng, J. Ma, and X. Yuan, "Deep learning for video compressive sensing," *APL Photonics*, vol. 5, no. 3, p. 030801, 2020.

[29] Z. Cheng, R. Lu, Z. Wang, H. Zhang, B. Chen, Z. Meng, and X. Yuan, "BIRNAT: Bidirectional recurrent neural networks with adversarial training for video snapshot compressive imaging," in *European Conference on Computer Vision (ECCV)*, August 2020.

[30] A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han, and G. Ding, "Yolov10: Real-time end-to-end object detection," *arXiv preprint arXiv:2405.14458*, 2024.

[31] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, "Yolov9: Learning what you want to learn using programmable gradient information," *arXiv preprint arXiv:2402.13616*, 2024.

[32] A. Farhadi and J. Redmon, "Yolov3: An incremental improvement," in *Computer vision and pattern recognition*, vol. 1804. Springer Berlin/Heidelberg, Germany, 2018, pp. 1–6.

[33] O. Shorinwa, J. Yu, T. Halsted, A. Koufos, and M. Schwager, "Distributed multi-target tracking for autonomous vehicle fleets," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 3495–3501.

[34] K. Fang, Y. Xiang, X. Li, and S. Savarese, "Recurrent autoregressive networks for online multi-object tracking," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2018, pp. 466–475.

[35] S. Sharma, J. A. Ansari, J. K. Murthy, and K. M. Krishna, "Beyond pixels: Leveraging geometry and shape cues for online multi-object tracking," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3508–3515.

[36] J. Xu, Y. Cao, Z. Zhang, and H. Hu, "Spatial-temporal relation networks for multi-object tracking," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 3988–3998.

[37] J. Ren, X. Chen, J. Liu, W. Sun, J. Pang, Q. Yan, Y.-W. Tai, and L. Xu, "Accurate single stage detector using recurrent rolling convolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2017, pp. 5420–5428.

[38] F. Yang, W. Choi, and Y. Lin, "Exploit all the layers: Fast and accurate CNN object detector with scale dependent pooling and cascaded rejection classifiers," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2016, pp. 2129–2137.

[39] N. Wojke and A. Bewley, "Deep cosine metric learning for person re-identification," in *2018 IEEE winter conference on applications of computer vision (WACV)*. IEEE, 2018, pp. 748–756.

[40] B. Veeramani, J. W. Raymond, and P. Chanda, "DeepSort: deep convolutional networks for sorting haploid maize seeds," *BMC bioinformatics*, vol. 19, no. 9, p. 289, 2018.

[41] X. Zhou, V. Koltun, and P. Krähenbühl, "Tracking objects as points," in *European Conference on Computer Vision (ECCV)*. Springer, 2020, pp. 474–490.

[42] P. Bergmann, T. Meinhardt, and L. Leal-Taixe, "Tracking without bells and whistles," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 941–951.

[43] K. Kang, H. Li, T. Xiao, W. Ouyang, J. Yan, X. Liu, and X. Wang, "Object detection in videos with tubelet proposal networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 727–735.

[44] X. Zhu, Y. Wang, J. Dai, L. Yuan, and Y. Wei, "Flow-guided feature aggregation for video object detection," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 408–417.

[45] K. Zhu, J. Dai, and Z. Gu, "Dynamic tracking method based on improved deepsort for electric vehicle," *World Electric Vehicle Journal*, vol. 15, no. 8, p. 374, 2024.

[46] X. Zhang, M. Qiao, L. Liu, Y. Xu, and W. Shi, "Collaborative cloud-edge computation for personalized driving behavior modeling," in *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing (SEC)*, 2019, pp. 209–221.

[47] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4377–4387, 2018.

[48] X. Zeng, B. Fang, H. Shen, and M. Zhang, "Distream: scaling live video analytics with workload-adaptive distributed edge intelligence," in *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, 2020, pp. 409–421.

[49] R. Bera, "Smart Automotive System With CV2X-Based Ad Hoc Communication," *Cloud and IoT-Based Vehicular Ad Hoc Networks*, pp. 293–323, 2021.

[50] M. Gonzalez-Martín, M. Sepulcre, R. Molina-Masegosa, and J. Gozalvez, "Analytical models of the performance of C-V2X mode 4 vehicular communications," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1155–1166, 2018.

[51] S. Fouladi, J. Emmons, E. Orbay, C. Wu, R. S. Wahby, and K. Winstein, "Salsify: Low-latency network video through tighter integration between a video codec and a transport protocol," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2018, pp. 267–282.

[52] A. Sapio, M. Canini, C.-Y. Ho, J. Nelson, P. Kalnis, C. Kim, A. Krishnamurthy, M. Moshref, D. R. Ports, and P. Richtárik, "Scaling distributed machine learning with in-network aggregation," in *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2021.

[53] R. Singh, S. Agarwal, M. Calder, and P. Bahl, "Cost-effective cloud edge traffic engineering with cascara." in *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2021, pp. 201–216.

[54] Y. Moon, S. Lee, M. A. Jamshed, and K. Park, "AccelTCP: Accelerating network applications with stateful TCP offloading," in *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2020, pp. 77–92.

[55] S. Lu, X. Yuan, and W. Shi, "Edge Compression: An integrated framework for compressive imaging processing on CAVs," in *2020 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 2020, pp. 125–138.

[56] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the h. 264/avc video coding standard," *IEEE Transactions on circuits and systems for video technology*, vol. 13, no. 7, pp. 560–576, 2003.

[57] A. Mercat, M. Viitanen, and J. Vanne, "Uvg dataset: 50/120fps 4k sequences for video codec analysis and development," in *Proceedings of the 11th ACM Multimedia Systems Conference*, 2020, pp. 297–302.

[58] D. Grois, D. Marpe, A. Mulayoff, B. Itzhaky, and O. Hadar, "Performance comparison of h. 265/mpeg-hevc, vp9, and h. 264/mpeg-avc encoders," in *2013 Picture Coding Symposium (PCS)*. IEEE, 2013, pp. 394–397.

[59] M. F. Duarte, M. A. Davenport, D. Takhar, J. N. Laska, T. Sun, K. F. Kelly, and R. G. Baraniuk, "Single-pixel imaging via compressive sampling," *IEEE signal processing magazine*, vol. 25, no. 2, pp. 83–91, 2008.

[60] G. M. Gibson, S. D. Johnson, and M. J. Padgett, "Single-pixel imaging 12 years on: a review," *Optics express*, vol. 28, no. 19, pp. 28 190–28 208, 2020.

[61] M. A. Davenport, M. F. Duarte, Y. C. Eldar, and G. Kutyniok, "Introduction to compressed sensing." 2012.

[62] Y. C. Eldar and G. Kutyniok, *Compressed sensing: theory and applications*. Cambridge university press, 2012.

[63] R. Huang, J. Pedoeem, and C. Chen, "YOLO-LITE: a real-time object detection algorithm optimized for non-GPU computers," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 2503–2510.

[64] M. Ghafoorian, C. Nugteren, N. Baka, O. Booij, and M. Hofmann, "El-gan: Embedding loss driven generative adversarial networks for lane detection," in *European Conference on Computer Vision*, 2018, pp. 256–272.

[65] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "BDD100K: a diverse driving dataset for heterogeneous multitask learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 2636–2645.

[66] M. Pitropov, D. E. Garcia, J. Rebello, M. Smart, C. Wang, K. Czarnecki, and S. Waslander, "Canadian adverse driving conditions dataset," *The International Journal of Robotics Research*, vol. 40, no. 4-5, pp. 681–690, 2021.

[67] Q. Luo, C. Li, T. Luan, and W. Shi, "Minimizing the delay and cost of computation offloading for vehicular edge computing," *IEEE Transactions on Services Computing*, 2021.

[68] Q. Luo, C. Li, T. H. Luan, and W. Shi, "Collaborative data scheduling for vehicular edge computing via deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9637–9650, 2020.

[69] M. Schwall, T. Daniel, T. Victor, F. Favaro, and H. Hohnhold, "Waymo public road safety performance data," *arXiv preprint arXiv:2011.00038*, 2020.

[70] Q. Wang and L. Zhou, "Research and application of j1939 protocol in engine parameter monitoring system," in *2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC)*. IEEE, 2020, pp. 1532–1534.

[71] T. M. Moerland, J. Broekens, and C. M. Jonker, "Model-based reinforcement learning: A survey," *arXiv preprint arXiv:2006.16712*, 2020.

[72] L. Liu, H. Lu, H. Zou, H. Xiong, Z. Cao, and C. Shen, "Weighing counts: Sequential crowd counting by reinforcement learning," in *European Conference on Computer Vision (ECCV)*. Springer, 2020, pp. 164–181.

[73] S. Lu, X. Yuan, A. K. Katsaggelos, and W. Shi, "Reinforcement learning for adaptive video compressive sensing," *ACM Transactions on Intelligent Systems and Technology*, vol. 14, no. 5, pp. 1–21, 2023.

[74] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 3645–3649.

[75] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960.

[76] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[77] Y. Liu, X. Yuan, J. Suo, D. J. Brady, and Q. Dai, "Rank minimization for snapshot compressive imaging," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 12, pp. 2990–3006, 2019.

[78] X. Yuan, "Generalized alternating projection based total variation minimization for compressive sensing," in *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2016, pp. 2539–2543.

[79] Y. Tian, K. Pei, S. Jana, and B. Ray, "DeepTest: Automated testing of deep-neural-network-driven autonomous cars," in *Proceedings of the 40th international conference on software engineering (ICSE)*, 2018, pp. 303–314.

[80] Y. Hou, Z. Ma, C. Liu, and C. C. Loy, "Learning lightweight lane detection CNNs by self attention distillation," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019, pp. 1013–1021.

[81] X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang, "Spatial as deep: Spatial CNN for traffic scene understanding," in *32nd AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[82] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[83] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[84] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[85] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 658–666.

[86] D. Poobathy and R. M. Chezian, "Edge detection operators: Peak signal to noise ratio based comparison," *IJ Image, Graphics and Signal Processing*, vol. 6, no. 10, pp. 55–61, 2014.

[87] S. Kumar, M. P. Andersen, H.-S. Kim, and D. E. Culler, "Performant TCP for low-power wireless networks," in *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2020, pp. 911–932.

[88] M. Qiao, Z. Meng, J. Ma, and X. Yuan, "Deep learning for video compressive sensing," *Apl Photonics*, vol. 5, no. 3, 2020.