EdgeCompression: An Integrated Framework for Compressive Imaging Processing on CAVs

Sidi Lu*, Xin Yuan† and Weisong Shi*
*Department of Computer Science, Wayne State University, Detroit, MI 48202, USA
†Nokia Bell Labs, Murray Hill, NJ 07974, USA
{lu.sidi, weisong}@wayne.edu, xyuan@bell-labs.com

***Abstract*—Machine vision is the key to the successful deployment of many Advanced Driver Assistant System (ADAS) / Automated Driving System (ADS) functions, which require accurate high-resolution video processing in a real-time manner. Conventional approaches are either to reduce the frame rate or reduce the related frame size of the conventional camera videos, which lead to undesired consequences such as losing informative high-speed information and/or small objects in the video frames.**

Unlike conventional cameras, Compressive Imaging (CI) cameras are the promising implications of Compressive Sensing, which is an emerging field with the revelation that the optical-domain compressed signal (a small number of linear projections of the original video image data) contains sufficient high-speed information for reconstruction and processing. Yet, CI cameras usually need complicated algorithms to retrieve the desired signal, leading to the corresponding high energy consumption. In this paper, we take a step further to the real applications of CI cameras in connected and autonomous vehicles (CAVs), with the primary goal of accelerating accurate video analysis and decreasing energy consumption. We propose a novel Vehicle-EdgeServer-Cloud *closed-loop* framework called EdgeCompression for CI processing on CAVs. Our comprehensive experiments with four public datasets demonstrate that the detection accuracy of the compressed video images (named *measurements*) generated by the CI camera is close to the accuracy on reconstructed videos and comparable to the true value, which paves the way of applying CI in CAVs. Finally, six important observations with supporting evidence and analysis are presented to provide practical implications for researchers and domain experts. The code to reproduce our results is available at https://www.thecarlab.org/outcomes/software.

***Index Terms*—Compressive imaging, Compressive sensing, Edge computing, Connected and autonomous vehicles, Deep learning, Convolutional neural networks.**

I. Introduction

The wide deployment of 4G/5G has enabled connected and autonomous vehicles (CAVs) as the perfect *edge computing* platforms [1] for a plethora of new intelligent services which were impossible before, such as obstacle avoidance and trajectory planning. These advanced services are built on top of a suite of sensors, including cameras, radars, and LiDARs. Among these sensors, cameras are one of the most essential parts because it is the critical component that enables real-time video frame analysis to detect the surrounding environment for accurate driver assistance and safe driving guidance. However, in a CAV, cameras could generate 20-60 MB data per second [2], and such vast volumes of data inevitably bring the challenges to the real-time analysis of the high-resolution video from an on-board camera. In this section, we list three main challenges faced by CAVs in terms of real-time video analysis.

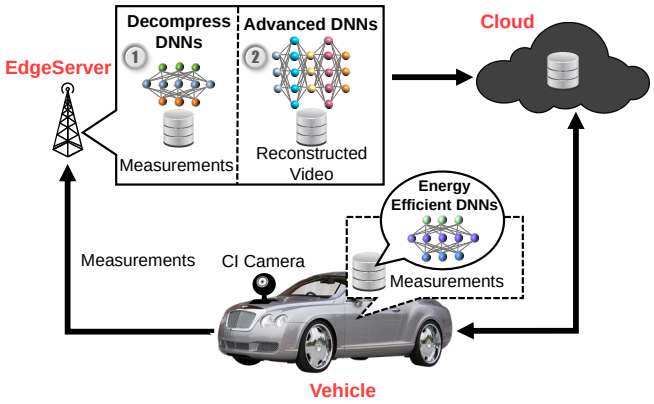

Fig. 1. The Vehicle-EdgeServer-Cloud *closed-loop* framework (dubbed EdgeCompression) to integrate compressive imaging (CI) and edge computing into CAVs systems.*"Measurements" refers to the optical-domain compressed video images*. The step of sending measurements from Vehicle to EdgeServer is asynchronous. A road side unit and cellular tower could both be treated as the EdgeServer.

*1) **Challenges in Imaging System**:* First, timely visual data analysis usually requires high-speed video collection from conventional cameras, which inevitably results in high bandwidth communication and power consumption. Current popular solutions to the problems of using conventional cameras are either to reduce the frame rate or reduce the related frame size. However, these two-fold solutions further lead to other undesired consequences, such as losing informative high-speed information and failing in small object detection in the video images.

*2) **Challenges in Neural Networks**:* Second, from the algorithm's perspective, although the state-of-the-art deep neural networks (DNNs) could achieve exceptional performance on *image* recognition tasks, it is nontrivial to transfer these DNNs to *videos* for real-time processing due to the slow per-frame inference and the high temporal redundancy [3]. Therefore, how to improve the efficiency of the visual processing for CAVs with a tolerable accuracy degradation has witnessed a tremendous amount of attention over the last few years.

*3) **Challenge in Edge Computing Platforms**:* Finally, traditional edge devices are usually resource-constrained with limited computation power and memory footprint. As a perfect

edge computing platform, although the computation resources of CAVs are becoming more and more powerful, they are still facing the challenge of providing accurate real-time services with stringent latency requirements under a limited energy and cost budget to process the increasing amounts of data.

Nowadays, there are three popular solutions to address the above challenges:

i) developing new imaging systems *i.e.,* developing novel Compressive Imaging (CI) cameras that are able to capture high-speed video information (described in Sec. I-A),

ii) developing advanced but light-weight algorithms (described in Sec. II-C1), and

iii) adopting hardware acceleration solution, such as involving the field-programmable gate array (FPGA) for video analysis tasks (described in Sec. II-C2).

While extensive researches in the edge computing field have been taken based on ii) and iii) [4]–[6], this paper first introduces CI camera [7], [8] into the edge-computing based CAV system for the purpose of achieving faster video processing, and less energy consumption and communication bandwidth. Moreover, we take one step further to solve the limitations of CI camera — requiring complicated algorithms to retrieve the desired signal and resulting in corresponding high energy consumption, by undertaking compressed video (measurement) analysis on the Vehicle directly and transfer the reconstruction workload to the EdgeServer with an event trigger to invoke necessary reconstruction.

The concise schematic of our proposed Vehicle-EdgeServer-Cloud *closed-loop* framework called EdgeCompression is shown in Fig. 1, where a compressive imaging camera is installed on a CAV to captured high dimensional (HD) information (rather than using a conventional camera). In this paper, *the (raw) measurement refers to the optical-domain compressed video, i.e., the raw data directly read out from the sensors*. As shown in Fig. 1, the raw measurements captured by the CI camera is on the one hand sent to the *energy-efficient DNNs* on the Vehicle to perform *real-time detection and reaction*. On the other hand, the raw measurements are sent to the EdgeServer to save the data and information. When it is necessary (*e.g.,* , a triggered event), a decompress DNN [9] is employed to perform the reconstruction (on the EdgeServer) to get the HD video and then using an advanced DNN to perform detection again, but this time on the reconstructed video. Besides, essential information and videos are sent to the Cloud for high-level planning and decision making and further refine the *energy-efficient DNNs* on the Cloud to update the DNN model on the Vehicle for accurate detection.

A. Compressive Imaging

Conventional digital imaging, since the invention of Charge-Coupled Device (CCD) [10], follows the procedure of projection, analog-to-digital conversion, and quantization to map the HD continuous information to two-dimensional discrete measurements. After decades of development, digital imaging has achieved great success and inspired a series of the related research field, such as machine vision, digital imaging processing, *etc.* This processing line and the associated image processing technologies, such as pattern recognition has greatly advanced the development of machine vision, which paves the foundation of CAVs. However, due to the inherent limitation of this "imaging first and processing afterwards" scheme, machine vision has encountered bottlenecks in both imaging and successive processing, especially in real applications such as CAV systems. The dilemma is mainly due to the limited information capture capability since during the imaging (capture), a large amount of information was filtered out (*e.g.,* spectrum, depth, broad dynamic range, *etc.*), and the sequence processing is thus highly ill-posed.

Compressive imaging or more generally computational imaging, by contrast, integrates the imaging system with consequent processing. This new scheme blurs the boundary between optical acquisition and computational processing, moves the calculation forward to the imaging procedure, *i.e.,* it introduces computing to design new task-specific systems. For instance, in some cases, CI changes the illumination and optical elements (including aperture, light path, sensor, *etc.*) to encode the visual information and then decode from the coded measurements computationally. The encoding can encode more visual information than conventional imaging systems or encode task-oriented information for better analysis.

In this paper, we use the video snapshot compressive imaging [9], [11]–[13] as an example to show the principle of our EdgeCompression *closed-loop* framework and demonstrate preliminary results considering the applications of CAV systems.

B. A New Closed-Loop Framework for CI Processing on CAVs

Recalling Fig. 1, the proposed EdgeCompression *closed-loop* framework consists of three primary parts: i) an *energy-efficient network* performing object detection directly on the compressed video (named measurement) captured by CI cameras on the Vehicle, to boost inference speed and reduce bandwidth, memory footprint, and energy consumption, ii) EdgeServer used to reconstruct the high-dimensional and high-speed data with a triggered event, instead of undertaking reconstruction on the Vehicle, so that it can transfer the energy pressure of reconstruction from the Vehicle to the EdgeServer and further reduce the unnecessary power consumption on the Edge Could with the help of the trigger; moreover, EdgeServer also works to verify the detection results of the Vehicle and send notifications and feedback to the Vehicle, and iii) Cloud that aggregates all useful information such as reconstructed video detection results and saved measurements from EdgeServer to refine the energy-efficient network on the Vehicle for model updating purpose. Besides, it also hosts all reconstructed videos of the vehicle fleets to further works such as traffic control and path planning.

It is worth noting that the "closed-loop" in our proposed framework is manifested in the following three aspects: a) the EdgeServer verifies the detection results from the Vehicle; b) the Cloud refines the network on the Vehicle, and c) the Cloud

collects all the information for traffic control and path planning and then sends navigation suggestions to the Vehicle.

C. Energy-Efficient Network

This line of machine vision-related research in the Edge Computing field is complemented by works of [4]–[6] where the focus is on either algorithm optimization or hardware acceleration to process images for object detection in a near real-time manner. For example, Masmoudi *et al.* [4] proved the capability of YOLOv3 (You Only Look Once-V3) [14] and Q-learning algorithm to handle real-time video frames processing for the autonomous vehicle-following problem. In the work of [5], Solovyev *et al.* implemented a higher number of convolutional blocks on an FPGA for the speed-up in video processing. Besides, Wei *et al.* [6] applied binarized neural networks on two FPGAs: PYNQ-Z1 and Zynq-Xc7Z010, which can realize a satisfying result in high analysis speed and accuracy to recognize pedestrians and some obstacles on a given road.

However, the valuable information stored in the HD data is usually overlooked. Thanks to CI, the HD data information can now be captured, but in a different manner. In this paper, we focus on accelerating video analysis speed by taking advantage of CI. Without loss of generality, we choose vehicle detection as our case study since it lays the foundation for the vital intelligent services in CAVs. The main idea is to map continuous multi-frame video images into *a single compressed measurement* [15]–[18], so that the application scenario of deep learning algorithms is transferred from the dense frames of videos to the informative *compressed measurements*. Following this method, the redundant computation could be reduced from the source. As such, the speed of model inference can be boosted significantly. Besides, it can also save the communication bandwidth and energy consumption due to the reduced computation workload. We believe "CI on Edges" will be the next trend of machine vision for CAVs.

D. Contributions of This Work

Our core contributions are not in the development of machine learning-based models that are built on top of well-understood and mature models, such as Faster RCNN [19], YOLOv3 [14], and Single Shot MultiBox Detector (SSD) [20]. Instead, the core innovation of our study is in providing experimental evidence to establish that raw measurements from CI cameras are effective in capturing the digital representation of the intricate features of targets. We also provide actionable insights on employing DNNs on measurements to speed up video analysis with high performance for intelligent services with stringent latency requirements meanwhile saving the memory, bandwidth, and energy consumption and thus potentially power and cost. Specific contributions are listed as follows.

- To the best of our knowledge, this is the first work that *introduces Compressive Imaging into Edge Computing for CAVs*. Our experiment results validate our assumption that building DNNs based on the measurements of the Vehicle is an effective solution to machine vision tasks — the accuracy of the detection on measurements are close to reconstructed videos, and their evaluation metrics are both comparable to the true value. Besides, it provides an alternative approach to speed up high-resolution video processing by multiple times for CAVs. Most importantly, this method can coexist with any existing algorithm optimization and/or hardware acceleration to further accelerate machine vision services. Besides, undertaking vehicle detection on raw measurements can also save communication bandwidth, memory, and reduce energy consumption thanks to the reduced computation tasks.
- We introduce EdgeServer that is responsible for critical two-fold tasks illustrated in detail in Sec. III-A: i) reconstructing HD data based on the measurements sent from Vehicle with a triggered event, instead of always undertaking reconstruction on the Vehicle. As such, it can transfer the energy pressure of reconstruction from the Vehicle to the EdgeServer that has efficient power and then conduct reconstruction when necessary. Consequently, it can also reduce the required power consumption of the Edge Could, ii) providing a complete verification closed-loop to guide the algorithm and system design of the Vehicle.
- The EdgeCompression *closed-loop* framework involves Cloud, where aggregating all the useful information from EdgeServer, including saved measurements and reconstructed video detection results (more accurate results), to refine the *energy-efficient network* and refresh it ON the Vehicle for the model updating purpose. Besides, it also hosts all reconstructed videos of the vehicle fleets, so it can further perform tasks which require the global view, *e.g.,* traffic control, and path planning.

The rest of this paper is organized as follows: Sec. II reviews related work and building blocks of the EdgeCompression *closed-loop* framework, which are detailed in Sec. III. Extensive experimental results are shown in Sec. IV and we present discussions in Sec. IV-D. Sec. V concludes the entire paper.

II. Background and Related Work

In this section, we review the essential building blocks of our new framework to integrate compressive imaging to edge computing with the application of CAVs. In particular, we describe the basic idea of video compressive imaging, recent advances of object detection based on deep learning, and neural networks for video compression. Research gaps are identified and inspire our own contributions.

A. Video Compressive Imaging

In video compressive imaging as shown in Fig. 2, the high-speed frames of a video are modulated at a higher speed than the capture rate of the camera. These modulated frames are then compressed into a single measurement. With knowledge of the modulation, multiple frames can be reconstructed from every single measurement. Therefore, CI was proposed to

capture high-dimensional data using low-dimensional detectors and then employ algorithms to solve such an ill-posed inversion problem. This leads to a *hardware encoder plus software decoder* regime, which is capable of providing us more information about the scene.

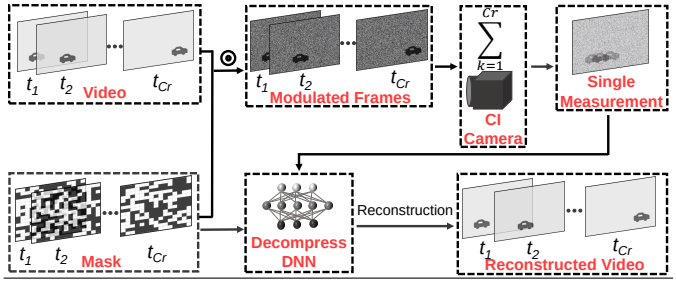

Fig. 2. The framework of video compressive imaging (CI). Here, $\odot$ denotes the element-wise product.

To be concrete, consider that Cr video frames (top-left in Fig. 2) are modulated by Cr different modulation patterns (bottom-left in Fig. 2) and then compressed to a single compressed measurement shown in the top-right of Fig. 2. Note the modulation patterns can be implemented by a translating mask [17] or other spatial light modulators [15], [16], [21], and both random binary patterns and gray-scale patterns can be used.

Let $\mathbf{X}_k \in \mathbb{R}^{N_x \times N_y}$ denote the k-th video frame, $\forall k = 1, \dots Cr$. During the CI capture, within one exposure time, each frame is modulated by a unique pattern $\mathbf{M}_k \in \mathbb{R}^{N_x \times N_y}, \forall k = 1, \dots, Cr$. These modulated frames (top-middle in Fig. 2) are then summed (integrating the light in the imaging system) to a single measurement $\mathbf{Y} \in \mathbb{R}^{N_x \times N_y}$. The forward model can thus be modeled as

$$\mathbf{Y} = \sum_{k=1}^{Cr} \mathbf{X}_k \odot \mathbf{M}_k + \mathbf{G}, \tag{1}$$

where $\odot$ denotes the element-wise product and $\mathbf{G}$ denotes the measurement noise. After the CI camera captures $\mathbf{Y}$, the next task usually is to perform the reconstruction, which aims to estimate $\mathbf{X}_k$ from $\mathbf{Y}$ given $\mathbf{M}_k$. It has recently been proved in [22] that, if the signal is structured enough, there exist reconstruction algorithms with bounded reconstruction error, even for $Cr > 1$.

During the past decade, though various algorithms have been developed [23]–[26], the long-running time of the algorithm [11] precludes wide applications of CI since in some cases, a real-time visualization is desired. Thanks to recent advances in deep learning, fast end-to-end reconstruction has been demonstrated in compressive imaging [12], [27]–[32]. The deep learning approach first *learns an approximate inverse function* of the system forward model in training, and then provides instantaneous reconstruction by directly estimating outputs from the input measurements during testing.

In a nutshell, CI and deep learning can now lead to an end-to-end high-speed capture, real-time and flexible reconstruction, which paves the way for real applications.

Different from the above "compressed capture and then reconstruction" research line, in this study, we aim to answer the following questions of using CI cameras in CAVs:

a) *Do we need to reconstruct high-quality data in real time on the vehicle?*
b) If the answer to a) is 'no', *can we still use CI cameras for real applications especially on CAVs?*

In this paper, we answer 'no' to a) but 'yes' to b). Specifically, we show that without reconstruction, object detection, which is essential in the real-world applications of CAVs, can be conducted *directly on the measurements* from CI cameras. This speeds up the detection speed with a comparable performance of ground truth, meanwhile saves the memory, bandwidth, and energy consumption. Furthermore, we *only perform reconstruction on the EdgeServer when it is necessary* such as surrounding vehicles are detected, rather than conducting reconstruction all the time on the Vehicle. This saves the energy on the Vehicle significantly. This new proposal lead to reals applications of CI to CAVs in an efficient way.

As objection detection is the primary task in CAVs, in the following, we review the recent advances of objection detection.

B. Milestones of Object Detection

Currently, the state-of-the-art technologies used for general object detection can be broadly categorized into three major classes: RCNN series [19], [33]–[35], SSD series [20], [36], and YOLO series [14], [37], [38].

1) Region-based Convolution Neural Network (RCNN): Before 2014, the mainstream method of object detection was the Deformable Part Model (DPM) [39], but it requires classifying an abundance of regions from the image. To bypass this problem, Girshick *et al.* proposed Regional Convolution Neural Network (RCNN) [34] that employs selective search to extract only two thousand regions named Region Proposal and run CNN on these limited regions — it was this time that CNN was first introduced to the object detection field, which greatly improves mean average precision (mAP). Based on RCNN, Fast RCNN [33] and Faster RCNN [19] were proposed to reduce the redundant computation for fast object detection applications.

2) You Only Look Once (YOLO): The YOLO series algorithm was firstly proposed by Redmon *et al.* in [37], and it is well known for its fast detection speed caused by simple and clear algorithm structure — YOLO formulates the object detection as a regression problem by solving a single CNN that predicts bounding boxes and category probabilities directly from images. The YOLO series algorithms have been continuously improved, and one of the most popular algorithms is YOLOv3 [14], which automatically selects the most suitable initial regression frame by incorporating the K-means clustering approach for a specific input dataset [40], [41].

3) Single Shot MultiBox Detector (SSD): To achieve a good balance between speed and accuracy, Liu *et al.* proposed SSD, which uses the regression method for object detection. Briefly

speaking, the SSD network was designed to replace the fully connected layer by the convolutional layer that is running on the input image only once and calculates a corresponding feature map.

C. Fast Object Detection

Although the trend of the image recognition tasks has been to make more complicated networks to achieve higher accuracy, these DNNs often required to be carried out in a timely manner even on resource-constrained devices in various real-world applications such as the autonomous driving considered in this work. An abundance of work has been published in this field to accelerate the inference of current DNNs so far. Next, we discuss a few prominent and representative approaches.

1) ***Technical Evolution on Model Optimization****:* References [42]–[44] aimed to develop light-weight DNNs with fewer parameters to achieve inference accelerating. For example, Iandola *et al.* [44] introduced SqueezeNet, a small CNN architecture. It could achieve AlexNet-level accuracy on ImageNet with $50\times$ fewer parameters and thus speeding up the inference time. Zhu *et al.* [45] introduced the deep feature flow for fast video recognition. In [46], Chen *et al.* explored a new method to speed up video recognition speed while maintaining competitive performance. Liu *et al.* [47] combined fast single-image detection with long short term memory networks to build a new online model for object detection in videos. Although their model was designed to run in real-time, the accuracy was much sacrificed.

2) ***Technical Evolution on Hardware Acceleration****:* In the meantime, significant efforts also focused on the efficient execution of DNNs over a constant stream of input video data. Most of these work used FPGAs to accelerate the inference speed since FPGAs could offer attractive solutions for both performance and power consumption. For example, in the work of [5], Solovyev *et al.* implemented a higher number of convolutional blocks on an FPGA for the acceleration in video processing. [48] introduced new design methods to speed up feature extraction algorithms on two types of FPGAs. Wei *et al.* [6] applied binarized neural networks on two FPGAs to realize a satisfying result in high speed and accuracy.

D. Deep Learning Model on Compressed Video

Most of the prior work focused on optimizing model architectures or involving FPGAs to improve the inference speed of DNNs. To our best knowledge, only a few prior efforts applied deep models directly on the compressed videos [47], [49], [50], [50] (but different from our video CI method). Briefly described, many multi-channel real-time systems have built-in VLSI hardware image compressors that can directly store compressed video data from multiple cameras to the hard disk. The main reason is that the uncompressed video (raw multi-channel video data) cannot be used due to the limitations of the available bus and processor. In the work of [3], [49], the video data was provided in a wavelet compression format, with the purpose of storing the image data in the file with as little space as possible. They utilized signals from compressed video to generate non-deep features, aiming to improve the speed and accuracy of video recognition. These studies are different from our work since they are still limited to conventional cameras and traditional compression approaches such as MPEG. By contrast, we employ a new imaging system (CI cameras), which is an optical (compressive) encoder itself into CAVs.

E. The Gap in Previous Work

Although the state-of-the-art DNNs could achieve outstanding performance on image recognition tasks, it is nontrivial to transfer these DNNs to videos for real-time processing due to the slow per-frame inference speed and the high temporal redundancy [3]. Moreover, all the above algorithms only win on either faster inference speed or higher recognition accuracy on a specific benchmark dataset (such as the COCO challenge [51]). For example, the YOLOv3-Tiny network [52], the fastest algorithm at present, achieves high inference speed at the expense of substantially lowered performance compared with other algorithms — the object detection in complex scenes is not accurate enough, *e.g.,* pedestrian objects are basically not detected in the road scene [53]. Moreover, hardware acceleration approaches could speed up image recognition, but it usually requires the involvement of specific hardware such as GPUs and/or FPGAs, which limits the application range and increases cost.

Bearing the above concerns in mind, we provide an alternative methodology to speed up the inference of DNNs for CAVs, which does not require any special hardware and can be applied on top of any state-of-the-art object detection algorithms to achieve further multiple times of acceleration based on the original inference speed. Specifically, we perform real-time object detection on the *measurements* directly captured by the CI cameras, without any further processing.

III. Proposed Framework and Methodology

A. EdgeCompression Closed-Loop Framework Description

Fig. 3 presents the detailed schematic of the designed EdgeCompression *closed-loop* framework. As mentioned in Sec. I-B, EdgeCompression includes three key components:

- i) An *energy-efficient network i.e.,* YOLOv3-Tiny [54] performs object detection directly on the raw measurement captured by CI cameras on the Vehicle, with the effectiveness of boosting inference speed, reducing communication bandwidth, memory requirements, and energy consumption.
- ii) EdgeServer reconstructs the HD data when the predefined event (such as vehicle detection) was triggered from the Vehicle, instead of undertaking reconstruction on the Vehicle, so that it can transfer the energy pressure of reconstruction from the Vehicle to the EdgeServer that possesses sufficient power. Besides, since reconstruction only occurs when the trigger is on, it will also reduce the required power consumption on the EdgeServer. Then EdgeServer leverages advanced network (YOLOv3) for

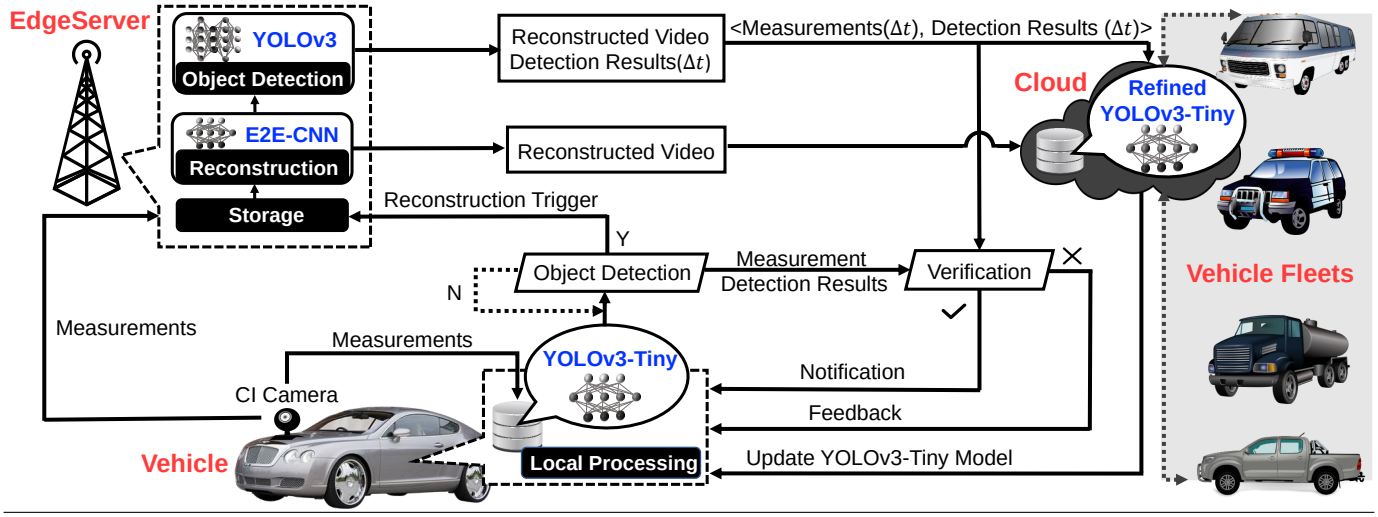

Fig. 3. The complete EdgeCompression *closed-loop* framework to integrate CI, Edge Computing, and CAVs. The CI camera is installed on the Vehicle to capture high-dimensional data, and a light-weight DNN (YOLOv3-Tiny) is employed to perform real-time detection based on raw measurements from the CI camera. The measurements (compressed thus low bandwidth) are also sent to the EdgeServer to save the information, and when the trigger is on, an end-to-end CNN (E2E-CNN) is employed to perform the reconstruction, and an advanced detection network (YOLOv3) is used to perform (more accurate) object detection based on the reconstructed video, which will also be used to verify the result of YOLOv3-Tiny. Meanwhile, these results will be sent to the Cloud with the saved measurements to refine YOLOv3-Tiny to provide more accurate on-vehicle services continuously. The reconstructed video will also be sent to the Cloud that received related information from Vehicle Fleets, to conduct decision making such as traffic control and path planning.

vehicle detection based on the reconstructed video. Moreover, EdgeServer also verifies the detection results of the Vehicle (thus closed-loop).

iii) Cloud aggregates all useful information such as reconstructed video detection results and saved measurements from EdgeServer to refine the YOLOv3-Tiny model on the Vehicle for the model updating purpose (again closed-loop). Besides, it also hosts all reconstructed videos of the vehicle fleets so that it can perform traffic control and path planning.

Vehicle: The installed CI camera of a Vehicle will continuously capture a constant stream of raw measurements of the surrounding environment. Meanwhile, an *energy-efficient model*, YOLOv3-Tiny, is running to detect vehicles based on the measurements simultaneously. YOLOv3-Tiny [54] is a light-weight deep learning algorithm designed for resource-constrained devices, with superior advantages on fast object detection due to the significantly reduced parameters.

In this EdgeCompression *closed-loop* framework, we assume one front CI camera is installed on the Vehicle. Without loss of generality, it is feasible to install multiple CI cameras to capture front, side, and back views and take place of the conventional cameras for real-world applications. Furthermore, as to the compression ratio (Cr), suppose that we collect 20 measurements per second at $Cr = 10$, we can obtain high-speed information of 200 HD frames per second after reconstruction on the EdgeServer, which is usually infeasible in conventional cameras. The frequency of collecting measurements could also be decreased based on the specific application requirements, *e.g.,* collecting 10 (or fewer) measurements per second, to save memory footprint, bandwidth, and energy. Therefore, employing CI cameras on Vehicles can not only capture and save high-speed information but also enjoys the flexibility of balancing bandwidth and energy.

From Vehicle to EdgeServer: There are two types of data transmission channels from a Vehicle to the EdgeServer. Firstly, the raw measurements captured by the CI camera will continuously be uploaded to the EdgeServer via a wireless network such as 4G or 5G. Secondly, once the surrounding vehicles are detected by the local YOLOv3-Tiny model, a reconstruction trigger will be sent to the EdgeServer to invoke video reconstruction for the stored raw measurements.

It is worth noting that since the CI camera is an *optical encoder*, the raw measurement is compressed already and thus the bandwidth to transmit the measurements to the EdgeServer will be small. More specifically, given a specific Cr, the bandwidth requirement of transmitting measurements from Vehicle to EdgeServer could be reduced to $\frac{1}{Cr}$ of the original bandwidth requirement, making the proposed closed-loop framework possible using 5G. Also, compared with a conventional camera, no additional processing such as image codec is required and thus potentially we can save the power and energy on the Vehicle. Furthermore, since the CI camera captures the HD data in a compressed manner, the information of all Cr frames are included in the measurement. Thus, HD data can be retrieved anytime when they are needed.

EdgeServer: As shown in Fig. 3, EdgeServer possesses two networks: E2E-CNN network [9] is responsible for the reconstruction, and it is able to provide remarkable performance for CI given sufficient training data. The other advanced network is working for vehicle detection based on the reconstructed video generated by E2E-CNN. Inspired by the evaluation results in [4], [55], which presented the effectiveness and efficiency of YOLOv3 and its capability to handle real-time

applications for CAVs, we choose YOLOv3 as the vehicle detection model on the EdgeServer.

Note that both reconstruction and advanced detection are performed only when surrounding vehicles are detected by the *energy-efficient network* (YOLOv3-Tiny) on the Vehicle. This saves the workload on the EdgeServer. Meanwhile, as mentioned above, the raw measurements from the CI cameras are saved on the EdgeServer, and the video can be reconstructed whenever it is necessary.

From EdgeServer to Cloud: On the EdgeServer, the YOLOv3 network takes the reconstructed video as the input and output vehicle detection results. Here, $\langle Measurements(\Delta t), DetectionResults(\Delta t)\rangle$ indicates a pair of input and output of EdgeServer during the time period of Δt. As to the data transmission, EdgeServer will continuously send $\langle Measurements(\Delta t), DetectionResults(\Delta t)\rangle$ to Cloud that possesses much more powerful computation resources to refine the YOLOv3-Tiny model (updating parameters for better detection). On the other hand, the reconstructed video will also be transmitted to Cloud with the goal of supporting future traffic control or path planning — we assume every CAV have equipped CI cameras and deployed our proposed EdgeCompression *closed-loop* framework, and following this method, Cloud will store the large-scale reconstructed video sent from different vehicles capturing diverse path conditions and traffic scenarios.

From Cloud to Vehicle: After Cloud finishes model refining, Vehicle will pull the refined YOLOv3-Tiny from EdgeServer to update the current network and providing more accurate detection results for the next round of detection.

From EdgeServer to Vehicle: On the EdgeServer side, since the YOLOv3 network should achieve higher accuracy than YOLOv3-Tiny with regard to vehicle detection in theory (we also verified this assumption in the experiment results of Section IV), we leverage the output — $DetectionResults(\Delta t)$ — as ground truth to verify whether the reconstruction trigger sent by Vehicle is correct or not. If the trigger is correct, it will send a related notification to the Vehicle. Otherwise, it will forward corresponding feedback to the Vehicle for model update purpose.

In the following, we select public datasets to verify our proposed EdgeCompression *closed-loop* framework. Before presenting the results, we introduce the datasets and hardware setup below.

B. Dataset Selection

The publicly available datasets have played a key role in pushing forward the development in many computed vision-based image analysis tasks by providing problem-specific examples and the corresponding ground truth [51], [56]. In the context of autonomous driving, the studies of [57]–[60] have provided the research community different benchmarks to evaluate diverse algorithms on the same data and contributed to closing the gap between the laboratory testing environment and various real-world problems.

Due to the methodology of mapping multiple images into a single measurement, the public datasets without a high frame rate or comprised of images collected at long intervals, such as Waymo [58] and KITTI [57] dataset, may not serve as an ideal evaluation benchmark for real-time vehicle detection on the measurements. In this work, the training and inference experiments are steered by four datasets, namely, AAU RainSnow Traffic Surveillance Dataset (AAU)[1], Berkeley DeepDrive BDD100K Video Dataset (BDD100K)[2], Public Dataset of Traffic Video (PDTV)[3], and DynTex Database (DynTex)[4].

AAU RainSnow dataset: AAU RainSnow dataset [61] contains 22 five-minute challenging sequences captured from seven traffic intersections in two cities, Aalborg and Vyborg, in Denmark. These video sequences collected from a conventional RGB camera in rainfall and snowfall under insufficient lighting conditions, with a resolution of 640 × 480 pixels at 20 frames per second. The glare from car headlights, reflections in puddles, and blurring of raindrops on the camera lens further weaken the visibility of the scene. Fig. 4(a) presents two typical examples of AAU video images.

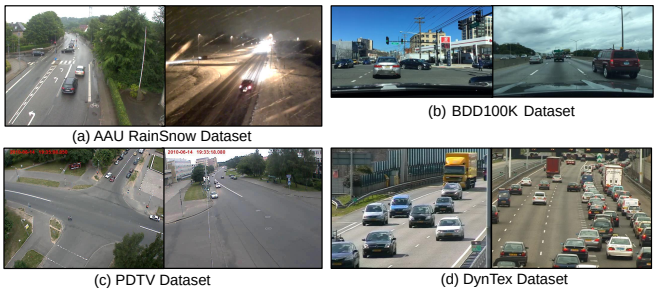

Fig. 4. Examples of four public video datasets.

BDD100K dataset: Berkeley DeepDrive BDDV100K dataset [62] is one of the largest real driving video datasets with 100K high definition videos over 1,100 hours of driving experience across different times in a day and weather conditions (shown in Fig. 4(b)). Each video is about 40 seconds long and with a resolution of 1280 × 720 pixels at 30 frames per second. The driving scenarios include highways, cities, and rural areas of several major cities in the United States.

PDTV dataset: The public dataset of traffic video (PDTV) [63] provides access to traffic videos of three intersections with annotations for real transportation applications, such as tracking road users and pedestrian infractions detection (shown in Fig. 4(c)). The video dataset was collected at three sites of Belarus and Canada with a resolution of 640 × 480 pixels at 20 and 40 frames per second, and the traffic scenes cross diverse traffic, lighting, and weather conditions.

DynTex dataset: DynTex dataset [64] is the first collection of high-quality dynamic texture videos that are structured

[1]https://www.kaggle.com/aalborguniversity/aau-rainsnow

[2]https://bdd-data.berkeley.edu/

[3]http://www.tft.lth.se/english/research/video-analysis/co-operation/public-dataset/

[4]http://dyntex.univ-lr.fr/database.html

by videos' underlying physical processes such as waving motion and discrete units, with the goal of serving as a standard database for dynamic texture research. More than 650 sequences of dynamic texture are available, and 9 sequences are related to traffic, with a resolution of 720 $\times$ 576 pixels at 25 frames per second (shown in Fig. 4(d)).

C. Hardware Setup

Recall the proposed EdgeCompression *closed-loop* framework hardware environment consists of the vehicle itself and an EdgeServer that has more powerful computing capabilities. In this paper, we adopt two types of hardware in total shown in Fig. 5 — Intel Fog Reference Design (FRD), and NVIDIA GPU Workstation. We assume that a CAV is equipped with an Intel FRD, and the NVIDIA GPU Workstation is working as the EdgeServer.

(a) (b)

Fig. 5. Two categories of hardware. Subfigure (a)-(b) shows the Intel Fog Reference Design (FRD), and NVIDIA GPU Workstation, respectively.

TABLE I
CONFIGURATION INFORMATION OF TWO HARDWARE DEVICES.

	Intel FRD	**NVIDIA GPU Workstation**
CPU	Intel Xeon E3-1275 v5	Intel Xeon E5-2690 v4
GPU	NONE	4 $\times$ 11 GB GeForce RTX 2080 Ti
Frequency	3.6 GHz	2.6 GHz
Core	4	14
Memory	32 GB	64 GB
OS	Ubuntu 16.04.6 LTS	Ubuntu 16.04.6 LTS

Intel FRD allows users to configure and program it for diverse use cases [65], and it is capable of providing a consistent throughput for various workload sizes [66]. In addition, as the powerful hardware with the high-quality components (4$\times$GeForce RTX 2080 Ti graphics cards), NVIDIA GPU Workstation is capable of delivering the cluster-level performance for even the demanding applications [67], [68]. We list detailed information about these edge devices in Table I.

IV. Experimental Design and Results Analysis

In this section, we present the experimental results of the proposed EdgeCompression *closed-loop* framework using the video CI and hardware described above on the four datasets.

A. Video Compression

Problem Statement. Given a selected CAV video dataset, *e.g.,* AAU RainSnow, we first compress the ground truth video to get the corresponding measurements, which is the simulated output of the CI camera, *i.e.,* mapping every Cr (compression ratio) frames into a single measurement. To compare the influence of different Cr's on the trade-off between detection accuracy and inference speed, we set $Cr = 6, 8, 10, 15$ for the selected four video datasets, respectively.

Recall the measurement model of video CI in Eq. (1), where the noise is considered. In our simulation, we ignore the noise term for simplicity, thus each measurement is now

$$\mathbf{Y} = \sum_{k=1}^{Cr} \mathbf{X}_k \odot \mathbf{M}_k. \tag{2}$$

The sensing mask $\{\mathbf{M}_k\}_{k=1}^{Cr}$ here is matrices comprised of 0 or 1 randomly, and it has the same size as the original video image. It can be seen that the measurement $\mathbf{Y}$ is a weighted ($\{\mathbf{M}_k\}_{k=1}^{Cr}$) summation of the high-speed frames $\{\mathbf{X}_k\}_{k=1}^{Cr}$. However, $\mathbf{Y}$ is usually a *non-energy-normalized* image. For example, some pixels in $\mathbf{Y}$ may gather only one- or two-pixel energy from $\{\mathbf{X}_k\}_{k=1}^{Cr}$, while some ones may gather $Cr - 1$ or Cr. Thus, it is not suitable to directly feed $\mathbf{Y}$ into a network, which motivates us to develop a measurement *energy normalization method* described below to make it feasible to be sent into the network.

To be concrete, we first sum all coding patterns $\{\mathbf{M}_k\}_{k=1}^{Cr}$ to achieve the energy normalization matrix $\tilde{\mathbf{M}}$ as

$$\tilde{\mathbf{M}} = \sum_{k=1}^{Cr} \mathbf{M}_k \,, \tag{3}$$

where each element in $\tilde{\mathbf{M}}$ describes how many corresponding pixels of $\{\mathbf{X}_k\}_{k=1}^{Cr}$ are integrated into the measurement $\mathbf{Y}$. We then normalize the measurement $\mathbf{Y}$ by $\tilde{\mathbf{M}}$ to obtain the energy-normalization measurement $\overline{\mathbf{Y}}$ as

$$\overline{\mathbf{Y}} = \mathbf{Y} \oslash \tilde{\mathbf{M}}, \tag{4}$$

where $\oslash$ denotes the matrix dot (element-wise) division. There is a small chance of some elements in $\tilde{\mathbf{M}}$ being zero and we calibrate them as one. Obviously, $\overline{\mathbf{Y}}$ owns more visual information than $\mathbf{Y}$. Meanwhile, $\overline{\mathbf{Y}}$ can be regarded as an approximate average of the original video frames $\{\mathbf{X}_k\}_{k=1}^{B}$, preserving the motionless information such as background and motion trail information.

This normalized measurement $\overline{\mathbf{Y}}$ can thus be sent to the network such as the YOLOv3-Tiny on the Vehicle. It is worth noting that $\tilde{\mathbf{M}}$ is fixed a video CI camera with a fixed Cr and thus it can be calculated in advance; therefore, our proposed energy normalization method only needs minimal computation (an element-wise division for each measurement) on the Vehicle and consumes limited resource.

Note that hereby we only considered the grey-scale video frames to demonstrate the idea. The RGB images with a Bayer

sensor can also be used but with some processing strategy as described in [18], [29].

Measurements Generation. Briefly, as to the AAU RainSnow dataset, we select 7 video segments and five minutes long of each. For BBD100K, 9 videos are chosen, and each video segment is about 40 seconds long. We then select 4 videos from PDTV, with 3.5 minutes collection period of each. Finally, with regard to DynTex dataset, we select 8 video segments related to vehicles, and the corresponding collection period is between 10 seconds and 27 seconds. Then we transfer these conventional RGB video images to the grey-scale images and compress these selected grey-scale images based on the Eq. (2), Eq. (3), and Eq. (4). We set $Cr = \{6, 8, 10, 15\}$ to test different compression ratios, and thus we obtain four groups of measurements for each dataset.

B. Video Reconstruction

Brief Decompress Model Background and Intuitions. Previous efforts proposed different algorithms to reconstruct high-speed video frames from a single measurement in the CI systems [11], [12], [22], [69]. In the latest work [9], Qiao *et al.* proposed the well-trained end-to-end convolutional neural network (E2E-CNN), which has led to significant improvements over existing algorithms — enabling a millisecond-level reconstruction for CI problems. Different from the conventional iteration-based algorithms [11] which must implement the iteration and computation for each measurement, the E2E-CNN conducts optimization only on the training phase and recovers images on the inference phase in an efficient way. Intuitively, E2E-CNN can provide video-rate high-quality reconstruction. In this work, we adopt this state-of-the-art algorithm as our reconstruction model to verify the applicability of vehicle detection on the measurements, and we train the E2E-CNN model based on the four selected video datasets.

To be specific, E2E-CNN takes the measurements and masks as input and outputs the reconstructed video. It consists of a convolutional encoder and decoder with Res-block connection [70] (Fig. 6(a)). In particular, there are five residual blocks in the encoder and decoder parts respectively, which are connected by two convolutional layers. The first convolutional layer is responsible for multi-dimensional feature extraction from the input data. After each convolution operation, ReLU activation and batch normalization are performed (Fig. 6(b)). In addition, the output of a encoder residual block is added ($\oplus$ to denote summation) to the input of the mapped decoder residual block. Moreover, E2E-CNN synthesizes the input of the network into the final reconstruction by leveraging large-span residual connection. In the network, it neither used upsampling nor pooling to avoid losing image details, and sigmoid was used as the activation function to ensure the desired scale of the final output.

E2E-CNN Training and Validation Methodology. We combine the compressed video segments from the four video datasets (described in Section III-B) for the training and testing purposes. We randomly take 80% of the measurements as the training dataset and the rest of the measurements for validation. We fit an E2E-CNN model on the training dataset and evaluate it on the validation dataset to access the performance of E2E-CNN on unseen measurements. Since not all these public datasets provide annotations, we directly employ open YOLOv3 network on the original public video dataset to get labels (bounding boxes of vehicles) and treat these labels as the ground truth.

To further strike the right balance between the compression ratio and the accuracy of the reconstruction, we have four experiment groups that compressed from the same video sets with different compression ratios, *i.e.,* $Cr = 6, 8, 10, 15$. Table II presents the change in the value of the loss functions (the mean square error) and PSNR [71] in the training and validation phases. Here, PSNR refers to the signal-to-noise ratio between two images, and we use it to evaluate the performance of E2E-CNN on image reconstruction.Let $\mathbf{X}^* \in \mathbb{R}^{N_x \times N_y \times Cr \times B}$ denote the ground truth video group, where B denotes the number of measurements being used, and $\hat{\mathbf{X}}$ be the reconstructed video from E2E-CNN with the same size of $\mathbf{X}^*$. The average PSNR of the video group is given by:

$$\text{PSNR} = \frac{\sum_{b=1}^{B}\sum_{k=1}^{Cr}\left[-10\log\frac{\sum_{n_x=1}^{N_x}\sum_{n_y=1}^{N_y}(\hat{x}_{n_x,n_y,k,b}-x^*_{n_x,n_y,k,b})^2}{N_x N_y}\right]}{B\cdot Cr}, \tag{5}$$

where $\hat{x}_{n_x,n_y,k,b}$ and $x^*_{n_x,n_y,k,b}$ denote the (n_x, n_y)-th pixel in the k-th frame of the b-th measurement in the estimated video and ground truth video, respectively. The higher the PSNR (less error), the better the quality of the compressed or reconstructed image.

Table II presents the change in the value of the training and validation loss functions and PSNR (dB) for four experiment groups whose $Cr = 6, 8, 10, 15$. As far as the loss function concerned, it can be seen that as the value of Cr increases, the value of loss increases, *i.e.,* the reconstruction accuracy decreases, which proves our conjecture — there is a trade-off between Cr (compression ratio) and the reconstruction accuracy. In addition, as to the PSNR value, we can see that when $Cr = 8$ is much closer to the loss value when $Cr = 10$. A higher or lower value of Cr leads to a bigger gap of two neighboring loss values. This indicates that the compression ratio between 8 and 10 might be a good choice for the selected video datasets to strike the right balance between compression ratio and reconstruction performance.

We also calculate the reconstruction time of a single compressed measurement and a single frame for four experiment groups shown in Fig. 7(a). Since the reconstruction time of a single measurement at $Cr = 10$ is less than 30ms, this proves the applicability of video-rate at 30 frames per second (FPS) reconstruction. Fig. 7(b) presents the exemplar reconstruction results with different compression ratios for the same scene.

C. Vehicle Detection Model

YOLOv3-Tiny Trained on Grey-Scale Video (TinyG). So far we have obtained three types of data *i.e.,*

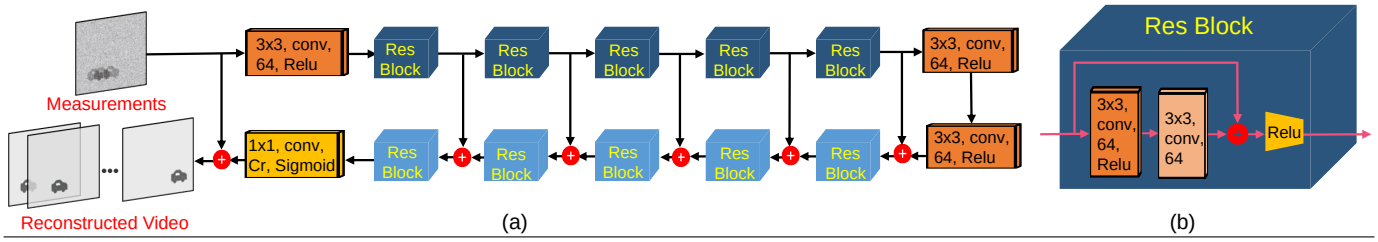

Fig. 6. E2E-CNN architecture (a) and Res-Block (b) used in E2E-CNN. $\oplus$ denotes summation.

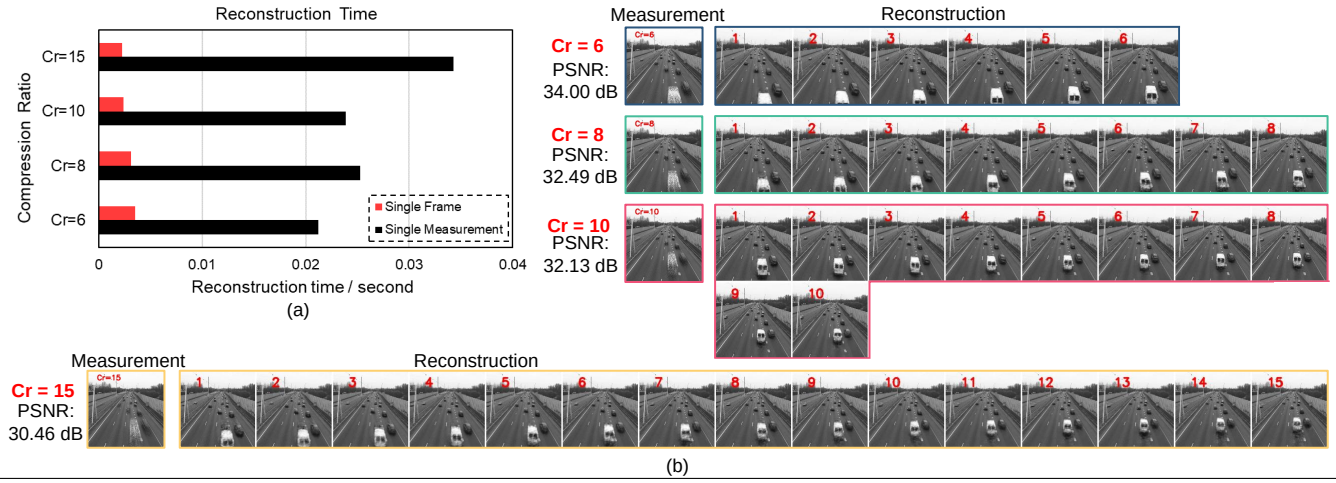

Fig. 7. (a): Reconstructed time of a single measurement and a single frame, (b): Reconstruction results with different compression ratio for the same scene.

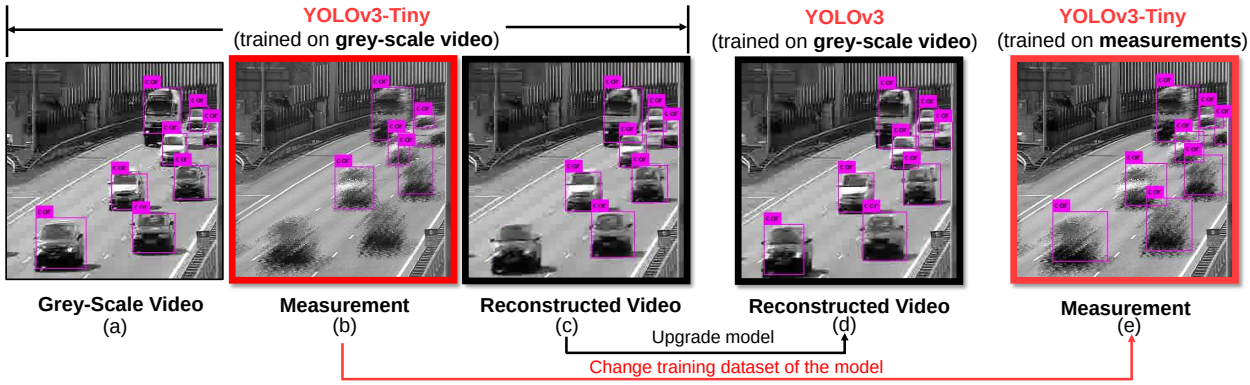

Fig. 8. Vehicle detection results on grey-scale video, measurements, and reconstructed video.

- grey-scale images (ground truth of E2E-CNN),
- measurements (input of E2E-CNN),
- the reconstructed video images (output of E2E-CNN).

Next, we aim to verify the effectiveness of measurements and reconstructed videos on the task of vehicle detection. To conduct a fair comparison, we plan to train the same model and employ this model on the above three types of data. In our proposed EdgeCompression *closed-loop* framework, measurements are captured by the CI camera of a Vehicle. On the Vehicle, apart from the accuracy and inference speed requirements, the DNN model should also be energy efficient. For this reason, we train YOLOv3-Tiny, a lightweight network, to detect vehicles based on the measurements directly (after the energy-normalization in Eq. (4)). Therefore, we also use the same trained model *i.e.,* YOLOv3-Tiny to detect vehicles based on grey-scale images and reconstructed images and treat the detection results of grey-scale images as the baseline to evaluate the effectiveness of measurements and reconstructed images in terms of vehicle detection. Fig. 8 presents the vehicle detection results on grey-scale video, measurements, and reconstructed video. In particular, Fig. 8(a)-(c) show the detection results of YOLOv3-Tiny that are trained on grey-scale images.

YOLOv3-Tiny Trained on Measurements (TinyM). Compare Fig. 8(b) with Fig. 8(a), the first two vehicles near the camera cannot be detected (shown in Fig. 8(b)), *i.e.,* the

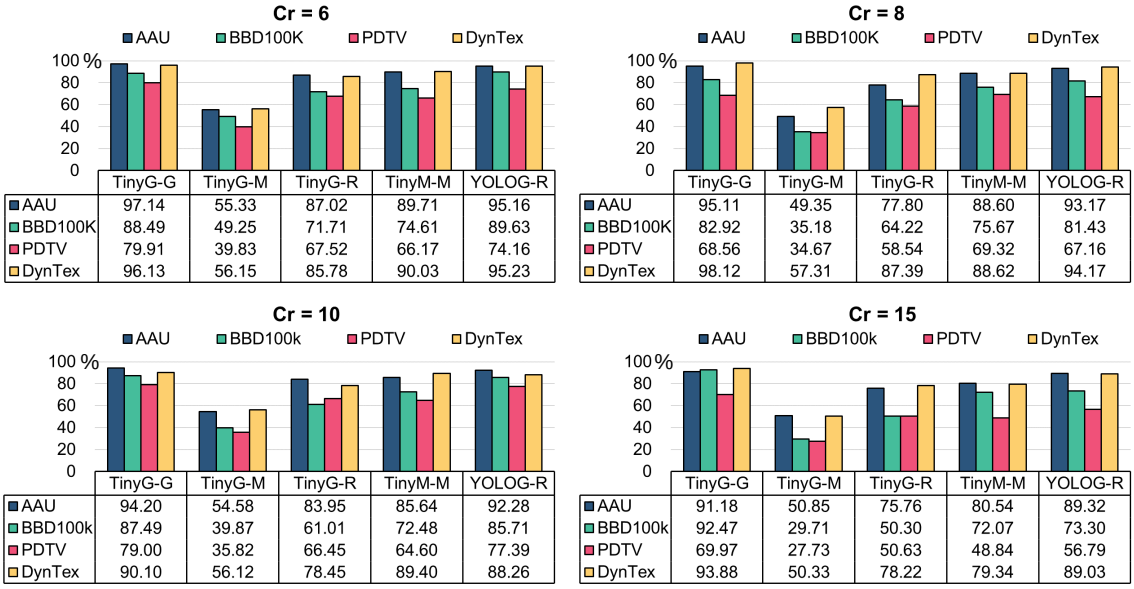

Fig. 9. Experiment results in terms of mAP. The letter (such as "G", "M", and "R") next to the model name indicates what dataset the model was trained on, and the letters behind the horizontal line indicate what dataset the trained model is used for vehicle detection.

TABLE II
THE CHANGE IN THE VALUE OF THE TRAINING AND VALIDATION LOSS FUNCTIONS AND PSNR (dB).

	Epoch		0	20	40	60	80	100
Loss	Cr=6	Train	0.106	0.046	0.041	0.039	0.039	0.039
		Valid	0.176	0.057	0.047	0.045	0.044	0.043
	Cr=8	Train	0.124	0.057	0.049	0.047	0.046	0.047
		Valid	0.120	0.073	0.049	0.045	0.045	0.044
	Cr=10	Train	0.080	0.054	0.050	0.050	0.050	0.049
		Valid	0.148	0.060	0.052	0.051	0.050	0.050
	Cr=15	Train	0.089	0.074	0.068	0.066	0.065	0.065
		Valid	0.143	0.075	0.067	0.067	0.066	0.065
PSNR (dB)	Cr=6	Train	25.05	31.95	33.02	33.50	33.76	33.75
		Valid	21.79	29.44	32.17	32.63	32.55	32.95
	Cr=8	Train	24.33	30.40	31.87	32.25	32.51	32.38
		Valid	24.51	28.52	32.32	32.81	33.06	32.99
	Cr=10	Train	27.50	31.21	31.86	32.09	32.04	32.19
		Valid	20.34	29.69	31.87	31.98	32.27	32.14
	Cr=15	Train	27.35	29.17	29.96	30.24	30.38	30.44
		Valid	21.12	29.90	30.91	30.67	30.91	31.13

YOLOv3-Tiny model trained on grey-scale image needs to be improved with regarding the detection accuracy when it is applied to the measurement. Inspired by this observation, we train a YOLOv3-Tiny network on the measurements directly.

Briefly, as to the four selected datasets, we have four sets of measurements due to the four compression ratios *i.e.,* $Cr = 6, 8, 10, 15$. Although the compression ratios are different, these measurements are still essentially images, but as Cr increases, these images become less and less clear. In this work, we combine all measurements as the input to train the YOLOv3-Tiny model, and we have the bounding box label (ground truth) annotated by the open YOLOv3 network, which enables model training. After training YOLOv3-Tiny on measurements, we conduct testing and obtain the new detection results (shown in Fig. 8(e)) — the first two vehicles can be detected successfully now.

YOLOv3 Trained on Grey-Scale Video (YOLOG). Similarly, compare Fig. 8(c) with Fig. 8(a), the first vehicle near the camera cannot be detected neither (shown in Fig. 8(c)). Hence, we consider upgrading the model *i.e.,* replacing the YOLOv3-Tiny model by YOLOv3, which usually achieves superior detection accuracy at the expense of substantially increased model size and computations. Since the reconstruction happens on the EdgeServer which has a higher computation capacity, deploying YOLOv3 on the EdgeServer with the purpose of detecting vehicles on the EdgeServer is thus a useful solution. It can be seen that in Fig. 8(d), the front vehicle can now be detected by YOLOv3, which was failed by YOLOv3-Tiny as in Fig. 8(c).

Effective Measurements. The detection result on the reconstructed video in Fig. 8(c) is more accurate than on the measurement in Fig. 8(b) but less accurate than the results of the grey-scale images in Fig. 8(a). Therefore, a question arise — whether measurements and reconstructed video can enable accurate vehicle detection in the real-time fashion? To answer this question, we then specifically focus on testing and evaluating the effectiveness of the raw measurements on the vehicle and the reconstructed video on the EdgeServer in a statistic manner. Towards this end, we employ mAP and FPS as the evaluation metrics. Here, mAP refers to the mean Average Precision, a widely used indicator to reflect the wellness of a DNN model in the object detection tasks [19]. FPS denotes the detection speed (inference speed) of DNNs, which can reflect the inference speed (the higher the better). Detailed observations and results are summarized in the next subsection. One example is shown in Fig. 8(e) that the front vehicle can be detected by training the YOLOv3-Tiny on the measurements instead of on the gray-scale video as in Fig. 8(b).

D. Results and Analysis

We now present and analyze the results of three DNN models — TinyG, TinyM, and YOLOG, in particular their

sensitivity toward three different inference datasets — grey-scale video, measurements, and reconstructed video that are obtained from the four datasets, and their limitations, and robustness. As far as the TinyG model concerned, *i.e.,* YOLOv3-Tiny model trained on grey-scale video, if it conducts the inference tasks on the unseen grey-scale video, we call it as "TinyG-G"; if it is tested on the measurements, we term it as "TinyG-M"; Similarly, we use "TinyG-R" to denote testing TinyG on the reconstructed video images. As far as the TinyM model concerned, *i.e.,* YOLOv3-Tiny model trained on measurements, we use "TinyM-M" to indicate testing TinyM on the measurements. Regarding the YOLOG model, *i.e.,* YOLOv3 model trained on grey-scale images, we focus on testing the performance of YOLOG on vehicle detection based on reconstructed videos, and we use "YOLOG-R" to denote it. In a nutshell, the letter (such as "G", "M", and "R") next to the model name indicates what dataset the model was trained on, and the letter behind the horizontal line indicates which dataset the trained model is used for vehicle detection.

The testing results are summarized in Fig. 9. First, it can be seen that the value of mAP is generally higher than the official testing results of YOLOv3 [14] (reported at https://pjreddie.com/darknet/yolo/). This is because we use open YOLOv3 to annotate bounding boxes of vehicles on the original RGB video dataset and treat these bounding boxes as the ground truth. Therefore, a portion of the vehicles that cannot be detected by these models also cannot be detected by YOLOv3 on the original RGB videos *i.e.,* these fail-to-detect vehicles have no negative impact on the evaluation accuracy of these models since they do not provide bounding boxes for the comparison purposes. We hope the results of the model detection are close to YOLOv3.

We present the key prediction quality measures for all models and feature sets (Fig. 9). Some interesting observations are listed follows, which include supporting evidence and reasons to explain the observed trends and practical implications for researchers and domain experts.

Observation 1: As to the vehicle detection from the measurements (TinyG-M and TinyM-M), training the model specifically on the raw measurement is necessary for the accurate detection since the mAP score of TinyM-M is significantly larger (almost double) than that of TinyG-M. In addition, as far as training the model from the grey-scale videos and then conducting inference based on the reconstructed video (TinyG-R and YOLOG-R), a more powerful model should be considered (YOLOG-R).

Observation 2: The vehicle detection results from the raw measurements (TinyM-M) achieve comparable detection results to the reconstructed video (TinyG-R), and they are close to the detection results from the ground truth video (TinyG-G) across all compression ratios, which answers the challenging questions proposed in Section II-A — we *do not need to reconstruct the high-quality data in real-time, and we can still use CI cameras for real applications in CAVs*.

Observation 3: With increasing Cr, the overall mAP across all application scenario decreases. For example, in DynTex, the mAP of TinyM-M decreased from 90.03% to 79.34% with an increase of Cr from 6 to 15. This confirms our hypothesis that a trade-off between mAP (accuracy) and the compression ratio should be balanced in real applications.

Observation 4: Comparing the results of these four datasets, we found that the vehicle detection on PDTV datasets has the overall lowest mAP (worst detection results). This is mainly because that the dataset was captured from the very high traffic cameras, which results in that vehicles are presented by a limited number of pixels in a video frame (examples in Fig. 4). In this challenging scene, the accuracy of vehicle detection would be affected adversely.

Observation 5: Regarding the AAU dataset with videos collected in rainfall and snowfall, vehicle detection models perform pretty well in the inference stage. This demonstrates the effectiveness of the presented model in terms of vehicle detection under challenging weather conditions.

Observation 6: As the representative of the video collected from the front camera of the vehicles *i.e.,* BBDK100, it achieves a relatively lower mAP than other normal traffic cameras *i.e.,* AAU and DynTex. The difference between the video collected from the in-vehicle camera and the traffic camera is — surrounding vehicles and the camera on the Vehicle usually move together, and the speed difference is often small. For the front camera, in addition to the inevitable shaking, it is highly possible to only see the rear of the front vehicles instead of the whole body in the captured video. Sometimes the vehicle in front is too big, and the in-vehicle camera can only capture a part of the rear of the vehicle, which increases the difficulty of detecting vehicles.

TABLE III
FPS OF THREE MODELS ON TWO HETEROGENEOUS HARDWARE PLATFORMS.

	YOLOG	**TinyG**	**TinyM**
NVIDIA GPU Workstation	67.55	336.93	334.11
Intel FRD	2.15	13.89	13.47

Table III shows the detection speed (in FPS) of the three models *i.e.,* YOLOG, TinyG, and TinyM on the two different hardware platforms *i.e.,* NVIDIA GPU Workstation and Intel FRD. It can be seen that the TinyG model has around 6× faster inference speed than YOLOG, which indicates the importance of applying a light-weight network to achieve near real-time detection. Besides, it is clear that in the Intel FRD that does not have a GPU, the detection speed is significantly lower than the NVIDIA GPU Workstation, even employing the state-of-the-art light-weight network YOLOv3-Tiny (TinyG and TinyM). In this context, the advantage of CI is becoming essential to speed up the inference of DNNs to achieve a $Cr\times$ acceleration.

E. Things We Tried That Did Not Work

During our experiments, we find some cases that object detection on the raw measurements fails to achieve high detection performance in terms of mAP, and we list them here for discussion. Firstly, as discussed in Sec. IV-D, when the vehicle is small, the vehicle detection from measurements suffers from some undesired missing detection. Secondly, when the color of the surrounding vehicles is close to the background, the effectiveness of vehicle detection from the measurements might be weakened. Thirdly, when the relative velocity between the camera and the targets is high, there is a probability of failure to detect vehicles on the measurement. This might due to that the change between the two adjacent frames will be relatively large, and when we compress the Cr frames together to generate the measurement, the trajectory shadow displayed by the vehicle will be longer than usual. On the other hand, due to the lack of sufficient training data capturing this special characteristic of long trajectory shadow, the TinyM model cannot learn this case and therefore cannot identify the vehicle. This observation indicates the need for sufficiently large training video sets before conducting the detection models on measurements.

V. Concluding Remarks

We have proposed a novel Vehicle-EdgeServer-Cloud *closed-loop* framework by integrating CI with edge computing in the application scenario of CAVs. To the best of our knowledge, this is the first work that provides an alternative method to achieve fast object detection by providing actionable insights on employing the state-of-the-art DNNs on measurements. Most importantly, this method can be employed on top of the popular fast DNN models and/or the hardware platforms that could speed up the inference of DNNs, such as FPGAs [72]. As such, the detection speed could be further boosted. Besides, both bandwidth, memory footprint, and energy consumption can be saved effectively due to the usage of the CI camera and the designed EdgeCompression *closed-loop* framework, which sheds the light on the real-world applications of CI into the CAV systems with edge computing technologies.

References

[1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.

[2] B. Krzanich, "Data is the new oil in the future of automated driving," *Intel Editorial*, 2016.

[3] C.-Y. Wu, M. Zaheer, H. Hu, R. Manmatha, A. J. Smola, and P. Krähenbühl, "Compressed video action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 6026–6035.

[4] M. Masmoudi, H. Ghazzai, M. Frikha, and Y. Massoud, "Autonomous car-following approach based on real-time video frames processing," in *2019 IEEE International Conference of Vehicular Electronics and Safety (ICVES)*. IEEE, 2019, pp. 1–6.

[5] R. Solovyev, A. Kustov, D. Telpukhov, V. Rukhlov, and A. Kalinin, "Fixed-point convolutional neural network for real-time video processing in FPGA," in *2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*. IEEE, 2019, pp. 1605–1611.

[6] K. Wei, K. Honda, and H. Amano, "FPGA design for autonomous vehicle driving using binarized neural networks," in *2018 International Conference on Field-Programmable Technology (FPT)*. IEEE, 2018, pp. 425–428.

[7] Y. Altmann, S. McLaughlin, M. J. Padgett, V. K. Goyal, A. O. Hero, and D. Faccio, "Quantum-inspired computational imaging," *Science*, vol. 361, no. 6403, 2018.

[8] J. N. Mait, G. W. Euliss, and R. A. Athale, "Computational imaging," *Adv. Opt. Photon.*, vol. 10, no. 2, pp. 409–483, Jun 2018.

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

[10] W. S. Boyle and G. E. Smith, "Charge coupled semiconductor devices," *The Bell System Technical Journal*, vol. 49, no. 4, pp. 587–593, 1970.

[11] 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.

[12] J. Ma, X.-Y. Liu, Z. Shou, and X. Yuan, "Deep tensor ADMM-Net for snapshot compressive imaging," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 10 223–10 232.

[13] X. Yuan, D. Brady, and A. K. Katsaggelos, "Snapshot compressive imaging: Theory, algorithms and applications," *IEEE Signal Processing Magazine*, 2020.

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

[15] Y. Hitomi, J. Gu, M. Gupta, T. Mitsunaga, and S. K. Nayar, "Video from a single coded exposure photograph using a learned over-complete dictionary," in *2011 International Conference on Computer Vision*, Nov 2011, pp. 287–294.

[16] D. Reddy, A. Veeraraghavan, and R. Chellappa, "P2C2: programmable pixel compressive camera for high speed imaging," in *CVPR 2011*, June 2011, pp. 329–336.

[17] P. Llull, X. Liao, X. Yuan, J. Yang, D. Kittle, L. Carin, G. Sapiro, and D. J. Brady, "Coded aperture compressive temporal imaging," *Opt. Express*, vol. 21, no. 9, pp. 10 526–10 545, May 2013.

[18] X. Yuan, P. Llull, X. Liao, J. Yang, D. J. Brady, G. Sapiro, and L. Carin, "Low-cost compressive sensing for color video and depth," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 3318–3325.

[19] 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.

[20] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

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

[22] 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.

[23] J. Yang, X. Yuan, X. Liao, P. Llull, G. Sapiro, D. J. Brady, and L. Carin, "Video compressive sensing using Gaussian mixture models," *IEEE Transaction on Image Processing*, vol. 23, no. 11, pp. 4863–4878, November 2014.

[24] J. Yang, X. Liao, X. Yuan, P. Llull, D. J. Brady, G. Sapiro, and L. Carin, "Compressive sensing by learning a Gaussian mixture model from measurements," *IEEE Transaction on Image Processing*, vol. 24, no. 1, pp. 106–119, January 2015.

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

[26] P. Yang, L. Kong, X. Liu, X. Yuan, and G. Chen, "Shearlet enhanced snapshot compressive imaging," *IEEE Transactions on Image Processing*, vol. 29, pp. 6466–6481, 2020.

[27] X. Yuan and Y. Pu, "Parallel lensless compressive imaging via deep convolutional neural networks," *Optics Express*, vol. 26, no. 2, pp. 1962–1977, Jan 2018.

[28] X. Miao, X. Yuan, Y. Pu, and V. Athitsos, "λ-net: Reconstruct hyperspectral images from a snapshot measurement," in *IEEE/CVF Conference on Computer Vision (ICCV)*, 2019.

[29] X. Yuan, Y. Liu, J. Suo, and Q. Dai, "Plug-and-Play algorithms for large-scale snapshot compressive imaging," in *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[30] Z. Cheng, R. Lu, Z. Wang, H. Zhang, B. Chen, Z. Meng, and X. Yuan, "BIRNAT: Bidirectional recurrent neural networks with adversarial train-

ing for video snapshot compressive imaging," in *European Conference on Computer Vision (ECCV)*, August 2020.
[31] Z. Meng, J. Ma, and X. Yuan, "End-to-end low cost compressive spectral imaging with spatial-spectral self-attention," in *European Conference on Computer Vision (ECCV)*, August 2020.
[32] Z. Meng, M. Qiao, J. Ma, Z. Yu, K. Xu, and X. Yuan, "Snapshot multispectral endomicroscopy," *Opt. Lett.*, vol. 45, no. 14, pp. 3897–3900, Jul 2020.
[33] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.
[34] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.
[35] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.
[36] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "DSSD: deconvolutional single shot detector," *arXiv preprint arXiv:1701.06659*, 2017.
[37] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.
[38] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.
[39] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1627–1645, 2009.
[40] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern recognition letters*, vol. 31, no. 8, pp. 651–666, 2010.
[41] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 2147–2154.
[42] Z. Wang, Z. Deng, and S. Wang, "Accelerating convolutional neural networks with dominant convolutional kernel and knowledge pre-regression," in *European Conference on Computer Vision*. Springer, 2016, pp. 533–548.
[43] 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.
[44] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50× fewer parameters and $<$ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.
[45] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei, "Deep feature flow for video recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2349–2358.
[46] K. Chen, J. Wang, S. Yang, X. Zhang, Y. Xiong, C. Change Loy, and D. Lin, "Optimizing video object detection via a scale-time lattice," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7814–7823.
[47] M. Liu and M. Zhu, "Mobile video object detection with temporally-aware feature maps," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5686–5695.
[48] O. Ulusel, C. Picardo, C. B. Harris, S. Reda, and R. I. Bahar, "Hardware acceleration of feature detection and description algorithms on low-power embedded platforms," in *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2016, pp. 1–9.
[49] B. U. Töreyin, A. E. Cetin, A. Aksay, and M. B. Akhan, "Moving object detection in wavelet compressed video," *Signal Processing: Image Communication*, vol. 20, no. 3, pp. 255–264, 2005.
[50] S. Wang, H. Lu, and Z. Deng, "Fast object detection in compressed video," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 7104–7113.
[51] 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.
[52] D. Xiao, F. Shan, Z. Li, B. T. Le, X. Liu, and X. Li, "A target detection model based on improved tiny-yolov3 under the environment of mining truck," *IEEE Access*, vol. 7, pp. 123 757–123 764, 2019.
[53] Z. Yi, S. Yongliang, and Z. Jun, "An improved tiny-yolov3 pedestrian detection algorithm," *Optik*, vol. 183, pp. 17–23, 2019.
[54] 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.
[55] B. Benjdira, T. Khursheed, A. Koubaa, A. Ammar, and K. Ouni, "Car detection using unmanned aerial vehicles: Comparison between Faster R-CNN and YOLOv3," in *2019 1st International Conference on Unmanned Vehicle Systems-Oman (UVS)*. IEEE, 2019, pp. 1–6.
[56] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.
[57] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3354–3361.
[58] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," *arXiv*, pp. arXiv–1912, 2019.
[59] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuScenes: A multimodal dataset for autonomous driving," *arXiv preprint arXiv:1903.11027*, 2019.
[60] S. Agarwal, A. Vora, G. Pandey, W. Williams, H. Kourous, and J. McBride, "Ford multi-AV seasonal dataset," *arXiv preprint arXiv:2003.07969*, 2020.
[61] C. H. Bahnsen and T. B. Moeslund, "Rain removal in traffic surveillance: Does it matter?" *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–18, 2018.
[62] 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," *arXiv: 1805.04687*, 2018.
[63] N. Saunier, H. Ardö, J.-P. Jodoin, A. Laureshyn, M. Nilsson, Å. Svensson, L. Miranda-Moreno, G.-A. Bilodeau, and K. Åström, "A public video dataset for road transportation applications," in *Transportation Research Board Annual Meeting Compendium of Papers*, 2014, pp. 14–2379.
[64] R. Péteri, S. Fazekas, and M. J. Huiskes, "DynTex: A comprehensive database of dynamic textures," *Pattern Recognition Letters*, vol. 31, no. 12, pp. 1627–1632, 2010.
[65] J. Li, J. Jin, D. Yuan, and H. Zhang, "Virtual fog: A virtualization enabled fog computing framework for internet of things," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 121–131, 2018.
[66] S. Biookaghazadeh, M. Zhao, and F. Ren, "Are FPGAs suitable for edge computing?" in *USENIX Workshop on Hot Topics in Edge Computing (HotEdge)*, 2018.
[67] F. Spiga and I. Girotto, "phiGEMM: a CPU-GPU library for porting quantum espresso on hybrid systems," in *2012 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing*. IEEE, 2012, pp. 368–375.
[68] I. V. Morozov, A. Kazennov, R. Bystryi, G. E. Norman, V. Pisarev, and V. V. Stegailov, "Molecular dynamics simulations of the relaxation processes in the condensed matter on GPUs," *Computer Physics Communications*, vol. 182, no. 9, pp. 1974–1978, 2011.
[69] X. Miao, X. Yuan, Y. Pu, and V. Athitsos, "λ-net: Reconstruct hyperspectral images from a snapshot measurement," in *IEEE/CVF Conference on Computer Vision (ICCV)*, vol. 1, 2019.
[70] 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*, 2016, pp. 770–778.
[71] 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.
[72] O. Consortium *et al.*, "OpenFog reference architecture for fog computing," *Architecture Working Group*, 2017.