Leveraging the edge and cloud for V2X-based real-time object detection in autonomous driving

Faisal Hawlader *, François Robinet, Raphaël Frank

Interdisciplinary Centre for Security, Reliability, and Trust (SnT), University of Luxembourg, L-1855, Luxembourg

ARTICLE INFO

ABSTRACT

Environmental perception is a key element of autonomous driving because the information received from the perception module influences core driving decisions. An outstanding challenge in real-time perception for autonomous driving lies in finding the best trade-off between detection quality and latency. Major constraints on both computation and power must be taken into account for real-time perception in autonomous vehicles. Larger detection models tend to produce the best results but are also slower at runtime. Since the most accurate detectors may not run in real-time locally, we investigate the possibility of offloading computation to edge and cloud platforms, which are less resource-constrained. We create a synthetic dataset to train object detection models and evaluate different offloading strategies. We measure inference and processing times for object detection on real hardware, and we rely on a network simulation framework to estimate data transmission latency. Our study compares different trade-offs between prediction quality and end-to-end delay. Following the existing literature, we aim to perform object detection at a rate of 20Hz. Since sending raw frames over the network implies additional transmission delays, we also explore the use of JPEG and H.265 compression at varying qualities and measure their impact on prediction. We show that models with adequate compression can be run in real-time on the edge/cloud while outperforming local detection performance.

1. Introduction

One of the core challenges in autonomous driving is to reliably and accurately perceive the environment around the vehicle. Perception is crucial to ensure safe driving because the information received from this task influences the core driving decision which determines how the vehicle should plan its path. However, perception requires processing a large amount of sensor data (*e.g.* camera, LiDAR, radar) in real-time. At the same time, the hardware embedded in vehicles is constrained by both cost and power consumption. Running all detection tasks locally can therefore require sacrifices on perception quality, in favor of real-time operation. One alternative is to offload some computations where resources are available. Compute capabilities are less limited on Multi-access Edge Computing (MEC) platforms [1], and the best hardware is available in the cloud. Offloading some computation to the cloud can be appropriated in some situations [2]. However, data offloading to MEC or cloud adds some additional transmission latency, which might not be acceptable for time-sensitive applications. The integration of C-V2X [3], supported by the latest 5G [4], presents an intriguing opportunity for data offloading option to improve perception in autonomous driving.

The idea behind data offloading is to keep a computationally intensive task in the MEC or cloud and perform lighter computations on the vehicle while maintaining high detection accuracy and respecting the latency requirements. However, data offloading can result in increased latency due to the time required to stream raw data between the vehicle and the MEC or cloud [5]. To address this potential increase in latency, we explore the use of H.265 [6], a video compression standard that exploits the temporal relationships between frames [7], and JPEG, a widely used method for image compression [8]. We evaluate these compression techniques at varying qualities, and we measure their impact on detection accuracy and end-to-end latency. Our investigation focuses on the integration of H.265 and JPEG streaming with edge and cloud platforms. By leveraging the computational capabilities of these platforms, we demonstrate how the proposed streaming solutions can be seamlessly integrated into the object detection pipeline, maximizing detection quality while minimizing latency. We follow the existing literature and aim to perform object detection at a rate of 20 Hz [9]. We investigate different variants of the YOLOv5 detection model [10], which offer different trade-offs between detection quality and inference time. As could be expected, Table 1 illustrates that larger models generally perform better. However, higher performance comes at the cost of increased computational requirements, hence adding additional

---

inference latency. Given the limitations of computing resources available in a vehicle, using larger models could result in latencies that exceed our 50 ms (20 Hz) requirements.

To evaluate the data offloading strategies proposed in this work, we employ a combination of real hardware and simulation techniques to comprehensively address the challenges in real-time object detection for autonomous driving. Specifically, we leverage the CARLA [11] simulator for environment rendering, camera sensor data generation, and real computing devices for object detection inference and processing time measurements. This approach allowed us to bridge the gap between simulation and real-world conditions, ensuring the practical relevance of our findings. Moreover, we have leveraged the OM-NeT++ [12], network simulation framework to accurately estimate network latency. This integrated approach, which combines simulation and real hardware, enhances the robustness of our research and the applicability of our findings to real-world scenarios. In this study, we are making the following contributions to the use of edge and cloud technology for real-time perception in autonomous vehicles:

- We collect a synthetic dataset to train and validate object detection models, utilizing the CARLA simulator [11]. This dataset serves as a valuable resource for our evaluation of the proposed offloading solutions. Furthermore, we contribute this synthetic dataset as an open-source asset to ensure reproducibility, facilitate ongoing research, and foster collaboration within the research community.[1]
- We investigate the transfer and processing of camera frames on edge and cloud platforms. Using real hardware setups, we measure processing latency, while data transmission latency is assessed through a network simulation framework. Subsequently, we analyze various trade-offs between prediction quality and end-to-end latency.
- To optimize transmission latency, we demonstrate a comprehensive framework that integrates H.265 and JPEG streaming using edge and cloud platforms for C-V2X based real-time object detection in autonomous driving. This solution addresses the challenge of transmitting camera sensor data with minimal latency while maintaining high-quality object detection.

The rest of the paper is organized as follows. In Section 2, we review the related literature. In Section 3, we describe the hardware used, our network simulation settings, and the training and evaluation of our object detectors. Section 4 presents the experimental results and discusses the different offloading trade-offs. Finally, in Section 5 we conclude this work with an outline of our contributions and a discussion of future work directions.

2. Related work

2.1. V2I communication

According to studies [13,14], autonomous vehicles typically leverage vehicle-to-infrastructure (V2I) communication technology that allows the vehicles to offload the task of sensor data processing to a dedicated server. The server could be placed on the edge using the 5G MEC [15] or in a cloud with higher computational resources [16]. V2I communications are carried out using the upload/download path [3]. The autonomous car offloads the task to an edge or cloud server that performs computations and sends the results back to the car. In this use case, the local device performs only mandatory pre-processing tasks, such as compression/encoding [17]. Offloading sensor data to the cloud through V2I introduces additional transmission latency [18]. According to [3], this communication latency can be reduced using MEC, which requires a lower transmission delay compared to the cloud. However,

Table 1
Average precision results for different model variants (AP@50). Model variants are detailed in Section 3.1.

	Model size		
	Small	Large	Large (high-res)
All (mAP)	0.64	0.66	0.85
Pedestrian	0.30	0.36	0.81
Traffic light	0.80	0.82	0.86
Vehicle	0.79	0.81	0.89

edge devices are also subject to limited resources that could limit application needs. These limitations may potentially restrict application requirements and, consequently, require further investigation to ensure compliance with latency constraints.

2.2. Object detection and impact of compression

Object detection: Pioneering work in object detection has used hand-crafted procedures to extract features from raw images, before using them as inputs to one or more object detectors. Popular examples include the Viola–Jones face detector based on Haar-like features [19], or the Histogram-of-Gradient detector [20]. Recent years have seen the emergence of two families of detectors based on deep neural networks: two-stage and single-stage models. Two-stage detectors first roughly identify regions that are likely to contain an object, before filtering and refining these object proposals with a trained model [21–23]. Although two-stage models achieve impressive accuracy, their computational complexity makes real-time operation a challenge. To remedy this, the SSD [24] and YOLO [25] family of models propose combining region proposal and refinement into a single operation. In YOLO, input frames are divided into cells and a set of bounding boxes are predicted for each cell. The YOLO detector can be trained end-to-end using a loss function that accounts for bounding box accuracy, objectness probabilities, and class assignments. Our work leverages YOLOv5 [10], a refined variant of the original YOLO method.

Impact of compression on detection: To enable faster data transmission for cloud inference, we study the impact of H.265 (video) and JPEG (image) compression on detection performance, for varying qualities. Existing work has shown that JPEG compression negatively affects the performance of models trained on uncompressed frames [26,27]. In the case of object detection, this effect is particularly noticeable at lower compression qualities. Performances deteriorate rapidly for moderate to heavy compression [26]. However, H.265 has recently gained significant attention because it offers superior video compression compared to its predecessor, making it an attractive choice for sensor data streaming. Although H.265 offers improved compression efficiency, high-quality video streams may still require significant bandwidth for transmission. Bandwidth limitations in edge-to-cloud communication or V2X networks may impact real-time streaming performance and object detection quality. Minimizing latency while maintaining the benefits of H.265 or JPEG compression poses a challenge that needs to be addressed.

2.3. Perception using C-V2X communication

Leveraging the advancements of 5G technologies, particularly edge and cloud computing, there is potential for C-V2X technology to be increasingly adopted and experimented [9,15,17]. However, widespread application of C-V2X remained mostly in the experimental phase, with larger-scale deployments expected to follow as the 5G infrastructure continues to evolve and mature. C-V2X qualifies to support advanced applications [28,29], such as collective perception [30]. Processing of perception sensor data using the onboard vehicle computer might not always be an option due to the limited computational power [16].

---

[1] **Dataset available at:** https://github.com/FaisalHawlader/V2X-Dataset.

Table 2
Based on inference time 50 ms constraints, we selected distinct platforms to run three models of different sizes. Fig. 1 compares inference times for different models and platforms.

Platform	Scenario/Model	Hardware configuration
Local ($\approx$20 W)	YOLOv5 small 157 layers, 7M params 640 × 640 Resolution	NVIDIA Jetson Xavier NX SoC Volta GPU, 384 CUDA cores Carmel ARMv8.2 CPU@1.9 GHz
Edge ($\approx$100 W)	YOLOv5 large 267 layers, 46M params 640 × 640 Resolution	Laptop with GeForce GTX 1650 Turing GPU, 896 CUDA cores Intel i9-9980HK @2.4 GHz
Cloud ($\approx$450 W)	YOLOv5 large high-res 346 layers, 76M params 1280 × 1280 Resolution	HPC node with Tesla V100 Volta GPU, 5120 CUDA cores Intel Xeon G6132 @2.6 GHz

Fig. 1. Model inference time comparison between different model sizes on different platforms. We use half-precision floating-point computation at inference time in order to speed up computation. The dashed line depicts the target 20 Hz latency for real-time object detection in autonomous driving scenarios.

However, collective perception using the V2X service allows cars to potentially offload sensory data to edge and cloud for resource-intensive computations [14,31]. Perception data processing in MEC or the cloud appears to be a viable option for autonomous driving, which has been the subject of many studies [3,14,16,17]. However, the perception of surrounding objects requires real-time detection, which demands rapid processing and low latency. This cannot be arbitrarily achieved, as it is heavily dependent on where the data is processed. In [30,32,33], different sensor data offloading strategies have been presented, ranging from raw data offloading to partially or completely processed data offloading to save network resources. However, their approach emphasizes data communication to reduce transmission overhead without evaluating the impact on perception quality. In principle, offloading raw sensor data is excellent for perception accuracy [34], but could increase transmission costs [35]. However, offloading compressed data can save network resources, but it might degrade detection quality [36]. To the best of our knowledge, the literature does not yet provide a study of the trade-off between detection quality and end-to-end processing.

3. Methodology

This section describes our initial efforts toward estimating the end-to-end delay of real-time object detection. Following existing work, our objective is to perform object detection at a rate of 20 Hz without degrading the detection quality [9]. In this context, the end-to-end delay heavily depends on where we perform the computation. If the car chooses to offload the data to the edge/cloud, it must exchange the raw data, which V2X technologies may not support due to bandwidth constraints [35].

3.1. Motivation & hardware

The limited computing resources and energy consumption make it difficult to detect objects in the car. Therefore, the computations are offloaded where the resources are available. However, reaching high detection quality and decreasing the inference delay remain challenging [37]. To better understand the problem and solve it, we performed a series of experiments on actual hardware setups. The hardware configurations are shown in Table 2, where local represents the onboard device of an autonomous car that has limited processing power. However, the compute capabilities are less constrained on edge platforms, and hardware only available in the cloud. This choice is made based on existing research [17,38]. To perform object detection, we use YOLOv5 [10], as it has been demonstrated to have superior performance in terms of accuracy and latency on state-of-the-art benchmark [39,40]. YOLOv5 offers various model sizes ranging from small to large. We observe that larger models are beneficial for detection quality (see Table 1), but there are inference timing constraints that must be taken into account. In this context, we aim to identify the best model for each platform that satisfies the time constraint. We use Fig. 1 to determine which model we can run on each platform. The inference time in Fig. 1 shows
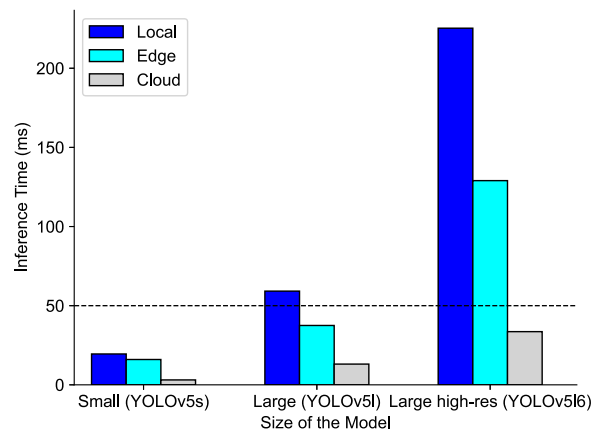
that the larger model has an inference time of more than 50 ms and as such is not suitable to run on the local platform. In view of the 20 Hz detection rate constraint, we decided to investigate the use of the small model on the local hardware, the large on the edge, and the large high-res one on the cloud. The exact YOLOv5 versions and the corresponding input resolutions are summarized in Table 2.

3.2. Networking aspects

In this section, we describe the main elements of the 5G Radio Access Network (RAN) and show how to use the network simulation framework to measure end-to-end network delays for a real-time object detection model supported by cloud infrastructure. To simulate the 5G data plane, we used Simu5G [3] which is a OMNeT++ based discrete event network simulation library [41]. It is worth noting that we specifically used release 16 of the standard. We focus on two scenarios, namely, perception data offloading with MEC and perception data offloading with cloud, leveraging the capabilities of C-V2X communication. The network environment we consider consists of a RAN and a 5G Core Network (CN). The RAN has a single 5G base station (BS), and one user equipment (UE) is attached to the BS, which is a vehicle in this use case [42]. Furthermore, we placed a MEC host at a distance of 500 m to the BS, establishing a wired network connection between them. This strategic placement ensures that the MEC system can closely interact with the BS, enabling rapid information exchange. The gNB is then connected to a cloud server via CN. The different components of the framework are shown in Fig. 2. We used a static setup in which the vehicle remained stationary throughout the experiments. We acknowledge, static setup is a limitation of our study, as it does not accurately reflect real-world scenarios where mobility is common. However, we believe that it does not significantly undermine the validity of our findings, particularly in relation to network latency, which is relatively small.

5G Core Network (CN): we consider a standalone version of the 5G core network [43], which meets our requirements and is also available for simulation. A Point-to-Point (PPP) network interface is used to connect gNB to the cloud through a wired connection [12]. The GPRS [18] tunneling protocol (GTP) is used to route IP datagrams (UDP) and establish a communication between gNB and the cloud.

Multi-access Edge Computing (MEC): The applications of MEC are growing and several standardization initiatives are being carried out to provide a successful integration of MEC into the 5G network [44]. This work considers a simplified MEC host-level architecture in accordance with the European Telecommunications Standards Institute
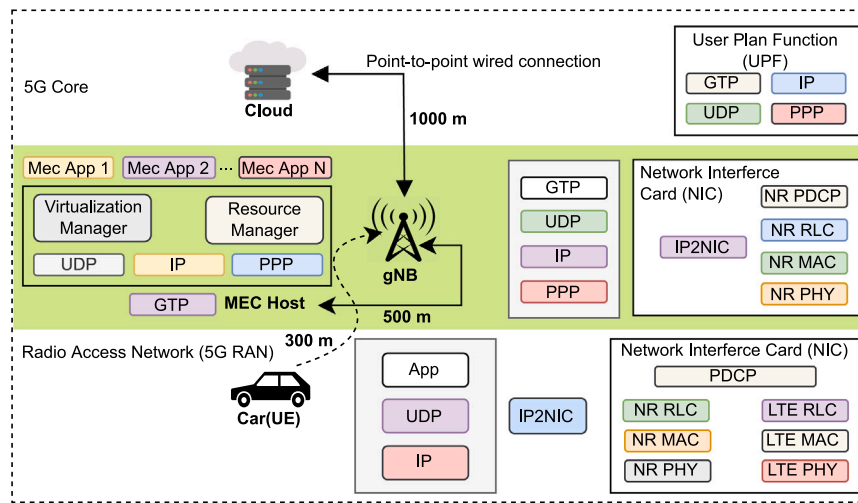
Fig. 2. Architecture of the end-to-end network simulation, showing the main elements of the 5G radio access network (RAN), including the multi-access edge computing (MEC) host-level components with a User Equipment (UE).

(ETSI) Ref. [15]. In our case, an MEC host is co-located with gNB as shown in Fig. 2. The MEC host provides various modules that allow MEC applications to operate efficiently and seamlessly. The MEC applications run in a virtual environment and the resource manager orchestrates the life cycle of those applications. The Virtualisation Manager allocates, manages, and releases virtualised aids such as computing, storage, and networking resources. The MEC host also includes a GTP protocol, which means that it can be located anywhere on the network. Following existing work, we placed the MEC 500 m away from the gNB [3]. A PPP wired connection with 100G data rate is used to connect the MEC to gNB [45].

5G base station (gNB): In the case of the scenario considered, gNB is configured with protocol up to Layer 3 and supports two network interface cards. One for PPP wired connectivity to connect the core network and the other for the radio access network. The internal structure of the two network cards is shown in Fig. 2. The PPP connection uses the GTP protocol, which has the same architecture as CN. On the other hand, the radio access network card has four modules. The topmost is the packet data convergence protocol (PDCP), which receives IP datagrams, performs ciphering, and sends them to the radio link control layer (RLC). RLC service data units are stored in the RLC buffer and retrieved by the underlying Media Access Control layer (MAC) when a transmission is required. The MAC layer aggregates the data into transport blocks, adds an MAC header, and sends everything through the physical layer (PHY) for transmission; for more details, we refer the reader to the Simu5G documentation [3].

User equipment (UE): As defined in the ETSI [15] and 3GPP specifications [46], user equipment is any device used by the end user. In our case, the user equipment refers to a car that is connected to the gNB, and equipped with C-V2X protocol stacks. We choose C-V2X because it is a 3GPP defined standard for connected mobility applications and works with 5G NR technology that is also available for simulation. It is important to mention that we used C-V2X only for bidirectional Vehicle-to-Infrastructure communications following the application needs. And the application we are focusing on is the offloading of perception data to the edge and cloud for real-time object detection. As part of the development policies and the implementation of Simu5G the UE has dual NIC to allow dual connectivity for both LTE [16] and 5G NR [29], as shown in Fig. 2.

3.3. Perception aspects

Dataset generation: In order to train the YOLOv5 models described in Table 2, we built a synthetic dataset using the CARLA simulator [11]. CARLA allows us to simulate a camera-equipped car driving

Table 3
Important network parameters with throughput and packet loss ratio for a stationary test. The averages are computed over 200 repetitions.

Parameter name	Value
Carrier frequency	3.6 GHz [47]
gNB Tx power	46 dBm
Path loss model	[3], Urban Macro (UMa)
Fading + Shadowing model	Enable, Long-normal distribution
Number of repetitions	200
Path loss model	3GPP-TR 36.873 [48]
UDP packet size	4096 B [49]
Throughput	113.94 Mbit/s (Avg.)
Numerology (μ)	3
Latency (Vehicle-to-Edge)	0.43 ms (Avg.)
Latency (Vehicle-to-Cloud)	0.45 ms (Avg.)
Packet loss ratio	0.0001

in a rendered town environment, specifically under different weather conditions. However, in this study, we only considered clear daylight weather conditions. Additionally, it provides access to ground truth bounding boxes for three classes of interest: vehicles, pedestrians, and traffic lights.

We run the simulation to collect ten thousand camera frames, taken at 1 Hz from the front of the ego-vehicle. We also collect ground truth 3D bounding boxes, which we project into camera frame coordinates to obtain 2D bounding boxes usable for training and evaluating YOLOv5. The dataset is split into three subsets: 6000 frames are used for training, 2000 for validation, and the remaining 2000 frames for testing. Table 4 presents the distribution of instances in different classes and provides essential information on the composition of the data set, where the first column denotes partitioning for different purposes. A notable aspect of this distribution is the considerably lower number of pedestrian instances compared to traffic lights and vehicles across all subsets. This imbalance may influence the performance of models trained on the dataset, potentially making them less proficient in detecting pedestrians compared to traffic lights and vehicles. Further analysis or data collection may be necessary to ensure representative and balanced data across all classes for more reliable results.

3.4. Exploring compression settings

We explore the influence of varying compression qualities on the volume of data to be transmitted to the edge or cloud. Our in depth analysis of various compression levels aims to identify the most

Table 4
Dataset composition and instance counts per class.

	Class		
	Pedestrians	Traffic lights	Vehicles
Train	12 916	43 418	33 351
Validation	2164	8272	11 295
Test	1756	11 115	7897
Total	16 836	62 805	52 576
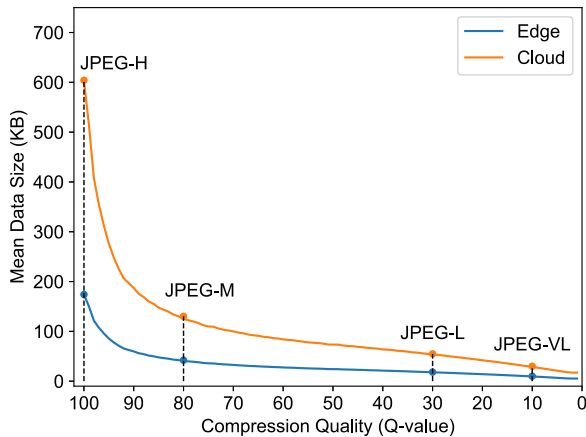
Fig. 3. The figure illustrates the mean data sizes for various JPEG compression qualities, namely JPEG-H (high quality), JPEG-M (medium quality), JPEG-L (low quality), and JPEG-VL (very low quality). We considered two distinct image dimensions: 640 × 640 for the edge and 1280 × 1280 for the cloud.

resource-efficient settings that can potentially optimize network resources and reduce the transmission latency, while preserving superior detection quality.

JPEG compression is a widely used compression algorithm known for its ability to strike a balance between image quality and file size reduction. It uses a lossy compression algorithm that selectively discards visual information to decrease file size. The compression level is largely controlled by a quality parameter, the *Q-value*, which ranges from 0 to 100. Higher *Q-values* preserve more details but result in larger sizes. In this study, we evaluated JPEG compression across a spectrum of quality levels from high quality (JPEG-H) to very low quality (JPEG-VL), while maintaining the default values for other parameters. The results, displayed in Fig. 3, demonstrate the mean data sizes associated with various JPEG qualities. For further investigation, we defined four distinct scenarios: JPEG-H (high quality) at *Q-value* 100, JPEG-M (medium quality) at 80, JPEG-L (low quality) at 30, and JPEG-VL (very low quality) at 10.

H.265 compression when using H.265 compression for camera sensor data streaming, several configuration and parameter settings can be adjusted. The selection of these parameters plays a pivotal role in navigating the balance between compression quality and latency. These settings include, but are not limited to, parameters such as Constant Rate Factor (CRF), presets, and lookahead settings, each offering various degrees of control over the encoding trade-off. The following discussion will elaborate on these parameters and their impact on the overall outcome.

- **H.265 Frames:** The fundamental computation of H.265 encoding starts with the estimation of three different frame types: I-frames (intra-coded frames), P-frames (predictive frames), and B-frames (bidirectional predictive frames). Each of these uniquely contributes to overall efficiency. The I-frame (known as the key frame) serves as the foundation, with all subsequent P-frame and B-frame relying on it. The principle of I-frames is based on the fact that neighboring pixels within an image often exhibit high

similarity. Minor differences between these adjacent pixels can be encoded using fewer bits, reducing the overall size.

The P-frames differ from the I-frames in that they are not self-contained. Instead, the P-frames contain only the changes in the stream from the previous frame. More specifically, a P-frame uses the prior I-frame or P-frame to encode the current frame, and hence making them predictive. They look at what has changed (such as the movement of objects) since the previous frame. If nothing has changed, no data need to be streamed, which is where much of the compression is achieved. If parts of the frame have changed, only the changes need to be encoded and streamed over the network.

However, B-frames refer to both the previous and future frames to achieve higher compression efficiency. They increase the encoding complexity and introduce some latency, but result in smaller file sizes and better quality. When H.265 is configured without B-frames, only I-frames and P-frames are used, reducing complexity and latency.

- **Lookahead** allows the encoder to examine future frames before encoding the current frame. This can increase compression efficiency and stream data quality, but can also introduce latency. If no lookahead is applied, this means that the encoder is not examining future frames before encoding the current frame. For the use case presented here, B-frames cannot be created since waiting for future information to encode the current frame implies an unacceptable increase of latency. Instead, stream compression may only rely on I-frames and P-frames. This setup is less efficient in terms of data compression compared to when B-frames are used, but it reduces latency and is suitable for real-time object detection in autonomous driving.

- **Constant Rate Factor (CRF):** strives to maintain a steady visual quality taking into account the complexity and motion within each frame. The H.265 compression assigns CRF values ranging from 0 to 51, which serve as quality-controlled variables affecting the bitrate. For instance, the CRF value can be adjusted to control the trade-off between quality and data size that needs to be transmitted to the edge/cloud. A lower CRF value gives higher quality but a larger data stream size. Conversely, a higher CRF value provides a smaller frame size but lower quality.

Using all possible CRF values, we performed a trade-off analysis to identify the optimal CRF for our real-time object detection use case. The lookahead functionality remained deactivated, and B-frames were excluded, while the rest of the parameters were set to their default values using FFmpeg [50]. In the absence of B-frames and lookahead in the encoding setup, a notable reduction in both computational complexity and latency was observed. These findings are particularly beneficial for applications that require real-time stream data transmission, where lower latency is a critical requirement. The results of these experiments, depicted in Fig. 4, illustrate the relationship between the CRF values and the resulting data size. Lower CRF values, which are indicative of higher quality, are correlated with larger data sizes. In contrast, increasing the CRF value leads to a reduction in the size of the data. This clearly demonstrates the trade-off between data size and the range of quality levels from high quality (H.265-H) to very low quality (H.265-VL). These observations will inform our decision-making on the quality levels that will be investigated further. For further investigation, we defined four distinct scenarios: H.265-H (high quality) at CRF 0, H.265-M (medium quality) at 24, H.265-L (low quality) at 30, and H.265-VL (very low quality) at 51. We considered two distinct image dimensions: 640 × 640 for the edge scenario and 1280 × 1280 for the cloud scenario. To facilitate a comprehensive evaluation, we categorized our analysis into four specific scenarios, each representing a different quality level in H.265 compression. We assigned CRF 0 to H.265-H for high quality, CRF 24 to H.265-M for medium, CRF 30 to H.265-L for low, and finally, CRF 51 to H.265-VL for very low quality, as demonstrated in Fig. 4.
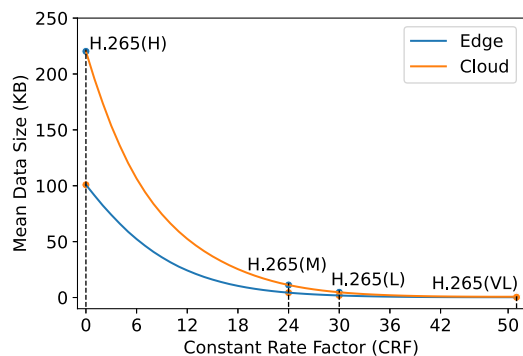
Fig. 4. Mean data sizes for various H.265 compression factor, namely H.265-H (high quality), H.265-M (medium quality), H.265-L (low quality), and H.265-VL (very low quality). We considered two distinct image dimensions: 640×640 for the edge scenario and 1280×1280 for the cloud scenario.

Training Protocol: All models are trained for 100 epochs on a single NVIDIA Tesla V100. We use the Adam optimizer with an initial learning rate of 0.001. In order to speed up training, we set the batch size to the maximal value that fits in GPU memory for each experiment.

4. Results

This section describes the experiments carried out to evaluate our proposed data offloading strategies. Experiments are performed utilizing the framework setup described in Table 2. Our analysis focuses on two critical key performance indicators, which are end-to-end delay and quality of detection. The close monitoring of these metrics enables the identification of potential bottlenecks or inefficiencies in real-time object detection for autonomous driving, ensuring that the vehicle responds accurately and in a timely manner to its surroundings.

4.1. End-to-end delay evaluation

To be able to compare the performance/latency trade-off of the three scenarios described in Section 3.1 and Table 2, we first measure their end-to-end delays.

In the case of the local platform, the delay depends only on the inference time on the onboard hardware. Note that non-maximum suppression (NMS) & input preprocessing are included in our inference time measurements, in addition to the forward pass of the model. By considering these integral components, we obtain a local end-to-end delay of 19.5 ms. This latency forms a critical part of our performance analysis, reflecting the efficiency of processing data locally without using any offloading strategies.

For the second and third scenarios, we evaluate the data offloading strategy between the car and edge or cloud platforms. Network latency is measured using the end-to-end simulation framework presented in Fig. 2. The important network simulation parameters with throughput and packet loss ratio are summarized in Table 3. In these situations, the end-to-end delay includes compression, transmission, decompression and inference. Although the time required to send the results back is not directly indicated, we assume an additional latency of 0.43 ms for the transmission delay, predicated on the notion that the raw detection results can be accommodated within a single packet. This assertion is supported by the data from Table 3, which shows that a single packet of size 4096 B takes approximately 0.43 ms (Avg.) to travel from the edge/cloud to the vehicle.

Sending raw uncompressed frames to remote platforms results in large transmission delays because of the size of the data. We measured an average end-to-end latency of 123.2 ms in the vehicle-to-edge scenario. This delay rises to 521.7 ms in the vehicle-to-cloud scenario due to the higher quality frame being processed by the cloud model. Since

these delays are not acceptable in most practical perception scenarios, we investigated the use of various compression strategies specified in Section 3.4 to reduce the volume of camera sensor data that needs to be transmitted to the edge or cloud over the network. Compression always occurs on the local device, and decompression happens either on the edge or cloud device, depending on the scenario. As illustrated in Table 5, compression can drastically reduce the size of the frame to be transmitted, allowing real-time remote object detection when C-V2X is available.

Table 5 illustrates a comprehensive analysis of the data size and end-to-end delay for edge and cloud scenarios under varying JPEG and H.265 compression qualities. When examining the edge scenario, the data size without compression is 1.23 MB, causing an end-to-end delay of 123.20 ms. The introduction of high quality JPEG compression significantly reduces the data size to 174.12 kB (*i.e.* 13.82% of the original), and the end-to-end delay accordingly decreases to 59.48 ms. As the quality of JPEG compression is further reduced, we see a concomitant decrease in both data size and end-to-end delay, reaching as low as 9.48 kB (or 0.75% of the original size) and a 37.27 ms delay with JPEG very Low compression. We also observed significant improvements with H.265 compression. By exploiting the temporal relationships between frames, H.265 is able to minimize the data size to 0.26 kB (or 0.01% of the original size), thus achieving a delay of 37.47 ms in a very low setting.

In the cloud scenario with no compression, the data size is considerably larger, starting at 4.92 MB with an end-to-end delay of 521.7 ms. However, similar to the edge scenario, the implementation of JPEG compression and H.265 compression shows a reduction in both the size of the data and the delay. With very low JPEG compression, the data size is reduced to 28.60 kB (or 0.6% of the original size) and results in a delay of 29.42 ms. With H.265 compression at a very low setting, the data size decreases to 0.69 kB (or 0.01% of the original) and the delay reduces to 27.78 ms. Table 5 conclusively demonstrates that the use of compression techniques can significantly reduce data size and latency in both vehicle-to-edge and vehicle-to-cloud scenarios. A breakdown and further details of these end-to-end delays are provided in Fig. 5. In all cases, compression and decompression have a negligible impact on the overall delay. As expected, network transmission increases with the amount of data to be transmitted. Inference latency are constant for a given platform, as the same model and input resolution are considered. In terms of end-to-end delay, all compression qualities are viable for real-time operation at 20 Hz on both platforms, except JPEG-H. The next section will investigate how compression impacts detection quality.
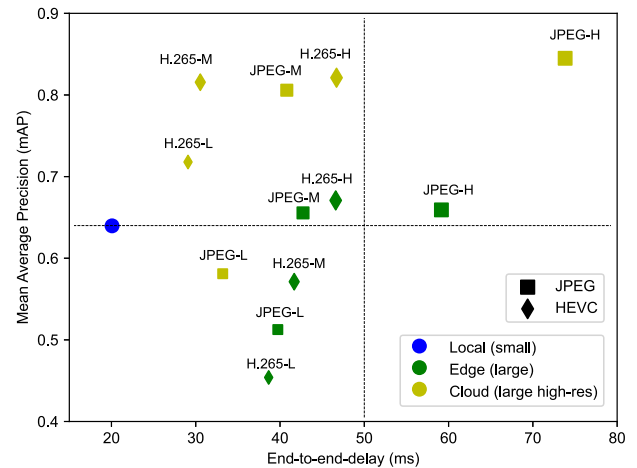
4.2. Analyzing detection quality

In order to obtain a complete picture of detection quality, measuring only Precision and Recall is insufficient. We follow the object detection literature and compute the Average Precision (AP), which is the area below the Precision-Recall curve. A detection is considered a true positive if its Intersection-over-Union (IoU) with a ground truth bounding box exceeds 50%. In order to derive a single metric for all classes, the per-class APs are averaged to obtain the Mean Average Precision (mAP).

As already discussed, the end-to-end delay can be significantly reduced using various compression techniques. However, excessive compression affects the detection quality, degrading the mAP. This section aims to determine the best trade-offs between detection quality and end-to-end latency. These trade-offs are illustrated in Fig. 6. As expected, the local operation is the fastest in terms of latency (19.5 ms) and obtains 64% mAP. As demonstrated in Fig. 6, the cloud platform consistently outperformed the edge platform in terms of mAP and end-to-end latency trade-off in all comparable compression techniques. For example, on the cloud platform, when using H.265-H compression, the mAP reaches 82%, but the end-to-end delay is
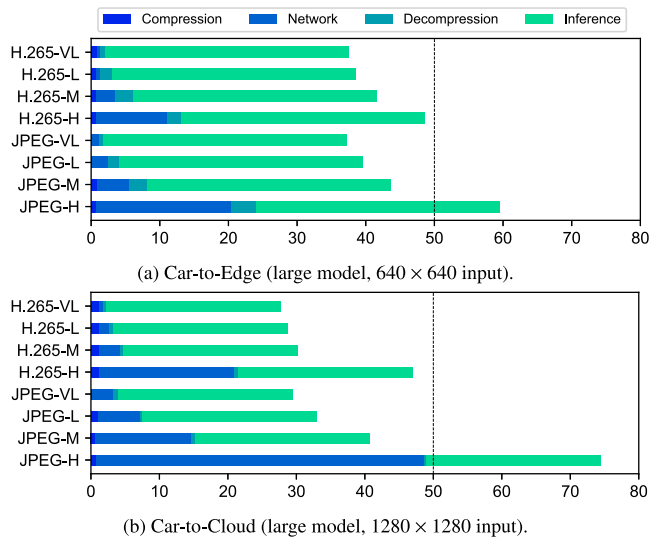
Table 5
Data size and end-to-end delay for both edge and cloud scenarios under different JPEG and H.265 compression qualities. The final column refers to the end-to-end delay in a vehicle-to-edge/cloud scenario. Fig. 5, illustrates a detailed breakdown of these delays.

Platform	JPEG quality	Avg. data size (% of original)	End-to-end delay (ms)
	No compression	1.23 MB (100%)	123.20
	JPEG-H	174.12 kB (13.82%)	59.48
	JPEG-M	40.78 kB (3.24%)	43.59
	JPEG-L	17.86 kB (1.42%)	39.62
Edge	JPEG-VL	9.48 kB (0.75%)	37.27
640×640	H.265-H	100 kB (2.00%)	48.65
	H.265-M	4.20 kB (0.09%)	41.61
	H.265-L	1.80 kB (0.04%)	38.51
	H.265-VL	0.26 kB (0.01%)	37.47
	No compression	4.92 MB (100%)	521.7
	JPEG-H	604.38 kB (12.00%)	74.50
	JPEG-M	125.51 kB (2.50%)	40.71
	JPEG-L	53.78 kB (1.13%)	32.93
Cloud	JPEG-VL	28.60 kB (0.6%)	29.42
1280×1280	H.265-H	220 kB (4.37%)	46.93
	H.265-M	11.20 kB (0.22%)	30.21
	H.265-L	4.69 kB (0.09%)	28.72
	H.265-VL	0.69 kB (0.01%)	27.78

(a) Car-to-Edge (large model, 640×640 input).

(b) Car-to-Cloud (large model, 1280×1280 input).

Fig. 5. Breakdown of average end-to-end delay (ms) into compression overhead, network transfer and inference times. The dashed line represents the 20 Hz latency constraint for real-time object detection.

Fig. 6. Trade-off between mean average precision (mAP) and end-to-end delay for different platforms and compression qualities. The end-to-end delay corresponds to the total of compression, network, decompression, and object detection inference delays. Very-low quality settings for both H.265-VL and JPEG-VL are not included, since their mAP fell below 10%, an unacceptable detection rate regardless of the reduced end-to-end delays.

46.93 ms. Interestingly, when using the H.265-M compression scenario, the mAP remains the same, but the delay is significantly reduced to 29.21 ms. Despite adjusting the CRF to 24 in H.265-M for enhanced compression, the mAP remained consistent at 82%, demonstrating that object detection performance was not compromised. The key revelation here is the substantial reduction in end-to-end delay, which plummeted to 29.21 ms. This can be attributed to the underlying principles of the CRF in the H.265 codec, which adeptly optimizes the trade-off between video quality and file size. Even with higher compression, essential object detection features and details are preserved. The observed reduction in file size led to improved data transmission efficiency, resulting in a remarkable decrease in end-to-end delay. In essence, the H.265-M compression scenario proves to be an attractive solution for real-time object detection on cloud platforms, where it maintained consistent detection accuracy while significantly improving responsiveness by reducing delay. Furthermore, lowering the compression quality to H.265-L slightly decreases the mAP to 72% and reduces the delay to

28.72 ms. Fig. 6 also shows an increase in mAP from 58% (JPEG-L) to 81% (JPEG-M), reaching 85% (JPEG-H) when considering JPEG compression techniques in the cloud. At the same time, the delay increases from 32.93 ms (JPEG-L) to 40.71 ms (JPEG-M) and peaks at 74.50 ms (JPEG-H). Although the highest detection quality (85% mAP) is achieved with JPEG-H compression, this comes at the cost of an increased end-to-end latency of 74.5 ms, which is not suitable for real-time object detection at 20 Hz. Nevertheless, it can still be used for applications where 10 Hz is an acceptable latency. For instance, in tasks that require long-range planning and decision-making, such as route planning and trajectory optimization, the slightly increased latency may not significantly impact overall system performance. These applications can still benefit from our analysis by strategically balancing detection quality and latency while utilizing Edge/Cloud, making it a suitable solution for real-world applications.

In the edge scenario, considering the H.265-High compression setting results in a mAP of 67%, accompanied by an end-to-end delay of 48.65 ms. For H.265-M and H.265-L, the mAP gradually decreases to 57% and 45%, with corresponding delay reductions of 41.61 and 38.51 ms. This trend is interesting, particularly in comparison to the cloud scenario. However, it aligns with previous investigations [51–53], which indicated that H.265 tends to perform better with high-resolution input images. In the JPEG case, the mAP increases from 51% (JPEG-L) to 66% (JPEG-M) and then to 67% (JPEG-H), while the delay increases from 39.62 ms (JPEG-L) to 43.59 ms (JPEG-M), reaching 59.48 ms (JPEG-H). Therefore, the edge scenario examined here is not advantageous over offloading to the cloud device. The use of better edge hardware specifically designed for mid-power inference rather than a traditional consumer GPU could most likely result in more competitive performance from the edge platform. On the other hand, the cloud platform is interesting, as it offers better performance with both JPEG and H.265, with 81% (JPEG-M) and 82% (H.265-H) mAP, respectively. Meanwhile, end-to-end delays are kept under 50 ms, respecting the 20 Hz constraint. Although compression is necessary for real-time operation on edge and cloud platforms, we observe its negative impact on detection quality. At extreme compression levels, remote detection mAP can drop below local performance while taking longer, rendering offloading harmful. For example, when using very low-quality settings for both H.265-VL and JPEG-VL in cloud or edge scenarios, the mAP dropped below 10%. This is considered unacceptable, regardless of the reduced end-to-end latency. We also

(a) H.265-H (CRF=0); Detected: 1 pedestrian, 7 vehicles, 8 traffic light

(b) H.265-M (CRF=24); Detected: 1 pedestrian, 7 vehicles, 7 traffic light

(c) H.265-L (CRF=30); Detected: 0 pedestrian, 7 vehicles, 6 traffic light

(d) H.265-VL (CRF=51); Detected: 0 pedestrian, 0 vehicles, 0 traffic light

Fig. 7. Visualization of the number of detected pedestrians, vehicles and traffic light in the cloud platform on different H.265 compression settings. A detection is considered only if its IoU with a ground truth bounding box exceeds 50%. Ground truth: 8 traffic lights, 8 vehicles and 2 pedestrians.

evaluate the detection quality separately for the three classes of interest: pedestrians, vehicles, and traffic lights. A visual representation is shown in Fig. 7. The purpose of this evaluation is to understand the impact of compression on different classes. The results are shown in Table 6. We observe that compression has a disproportionate impact on Average Precision (AP) for classes that are typically smaller in scale. We benchmark the performance of various compression settings with the corresponding platforms against the baseline scenario, *i.e.* the local platform operating without any compression. The local platform performance shows an AP of 30% for pedestrian detection and achieves an AP of 79% and 80% for vehicle and traffic light detection, respectively. The Table 6 demonstrate that the use of a cloud platform, compared to local and edge platforms, led to significant performance improvements for pedestrian detection, particularly with medium compression settings for both JPEG-M (+116%) and H.265-M (+130%). In particular, high-quality compression (JPEG-H and H.265-H) enables both edge and cloud platforms to outperform the local scenario, with a cloud platform improving pedestrian detection by 166% (JPEG-H) and 126% (H.265-H) and an edge platform achieving a 20% increase in AP in both cases. Although all platforms perform similarly for vehicle and traffic light detection with high and medium compression settings, the performance decreases noticeably at lower compression settings. Under very low quality compression (JPEG-VL and H.265-VL), the AP for all three classes drops on both platforms, underlining the limitations of excessive compression. These findings underline the potential of edge and cloud platforms, paired with appropriate compression settings, to improve object detection performance relative to a local platform, especially for pedestrian detection. However, the benefits diminish with lower compression quality, emphasizing the need to strike the right balance between compression level and detection performance for real-time object detection in autonomous driving.

5. Conclusion and future work

In this work, we have explored the possibility of real-time remote object detection. Although larger models perform better, they also require higher computational power. Considering cost and power constraints in autonomous vehicles, the very best models cannot run locally in real-time. To solve this problem, we have proposed different strategies to offload object detection to edge or cloud devices using C-V2X. We have compared these strategies in terms of their detection quality

Table 6

Per-class AP for different compression qualities and platforms. The input qualities are 640 × 640 for the local and edge platforms, and 1280 × 1280 for the cloud model. The baseline local model is highlighted in gray color, while all the models that are competitive, meeting both the mAP and latency constraint, are bolded.

Compression	Platform	Pedestrian (+% of local)	Vehicle (+% of local)	Traffic light (+% of local)
No compression	Local	0.30 (0%)	0.79 (0%)	0.80 (0%)
	Edge	0.36 (+20%)	0.81 (+2%)	0.82 (+2%)
	Cloud	0.81 (+170%)	0.89 (+12%)	0.86 (+7%)
JPEG-H	Edge	0.36 (+20%)	0.80 (+1%)	0.82 (+2%)
	Cloud	0.80 (+166%)	0.89 (+12%)	0.86 (+7%)
JPEG-M	Edge	**0.41 (+36%)**	**0.83 (+5%)**	**0.78 (−2%)**
	Cloud	**0.65 (+116%)**	**0.88 (+11%)**	**0.85 (+6%)**
JPEG-L	Edge	0.35 (+16%)	0.77 (−2%)	0.74 (−7%)
	Cloud	0.43 (+43%)	0.84 (+6%)	0.81 (+1%)
JPEG-VL	Edge	0.23 (−23%)	0.73 (−7%)	0.55 (−31%)
	Cloud	0.24 (−20.00%)	0.78 (−1%)	0.62 (−22%)
H.265-H	Edge	**0.36 (+20%)**	**0.80 (+1%)**	**0.83 (+3%)**
	Cloud	**0.68 (+126%)**	**0.83 (+5%)**	**0.83 (+3%)**
H.265-M	Edge	0.25 (−16%)	0.78 (−1%)	0.82 (+2%)
	Cloud	**0.69 (+130%)**	**0.84 (+6%)**	**0.82 (+2%)**
H.265-L	Edge	0.04 (−86%)	0.59 (−25%)	0.72 (−10%)
	Cloud	**0.51 (+70%)**	**0.82 (+3%)**	**0.82 (+2%)**
H.265-VL	Edge	0.01 (−96%)	0.02 (−97%)	0.03 (−96%)
	Cloud	0.03 (−90%)	0.23 (−70%)	0.04 (−95%)

and compliance with end-to-end latency requirements. To evaluate the proposed strategies, we have generated a synthetic dataset and have trained different variants of the YOLOv5 architecture. Using an end-to-end 5G network simulation framework, we have measured the network latency incurred when transferring camera frames for processing on the edge and cloud. We have also analyzed how the use of heavy compression can reduce the frame size by up to 98% when using JPEG and H.265 to enable real-time remote processing.

We showed that models with adequate compression can be run in real-time on the edge/cloud while outperforming local detection performance. The experimental results demonstrated that the H.265 (video) compression technique generally offers better performance in terms of detection quality and end-to-end latency trade-off compared to JPEG (image), particularly in the cloud scenario. However, there are

scenarios where JPEG compression is still sufficient and can be used, such as where 10 Hz is an acceptable latency. Our experimental results show that excessive compression affects the detection quality compared to raw frames, particularly for the pedestrian class.

Future work will focus on testing the offloading strategies in different driving environments. Since local perception is still needed as a fallback to cope with bad connectivity, we plan on investigating the impact of mode switching between local and remote processing on detection quality and latency. Additionally, we will examine the influence of mobility on network latency, considering factors such as signal strength, handovers, obstacles, interference, and network congestion. Furthermore, we aim to evaluate the scalability of our system beyond a single base station and UE. This future work will encompass scenarios involving multiple base stations and UEs, with a specific focus on cooperative perception solutions. We intend to explore a range of configurations to provide a comprehensive understanding of how our system behaves under different loads and network conditions as it scales. We believe that such an analysis will provide valuable insights into the broader applicability and performance validity of our approach.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Faisal HAWLADER reports financial support was provided by National Research Fund Luxembourg (FNR). If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

References

[1] F. Hawlader, F. Robinet, R. Frank, Vehicle-to-infrastructure communication for real-time object detection in autonomous driving, in: 2023 18th Wireless on-Demand Network Systems and Services Conference (WONS), IEEE, 2023, pp. 40–46.

[2] A.B. De Souza, P.A. Rego, T. Carneiro, J.D.C. Rodrigues, P.P. Reboucas Filho, J.N. De Souza, V. Chamola, V.H.C. De Albuquerque, B. Sikdar, Computation offloading for vehicular environments: A survey, IEEE Access 8 (2020) 198214–198243.

[3] G. Nardini, G. Stea, A. Virdis, D. Sabella, Simu5G: a system-level simulator for 5G networks, in: SIMULTECH 2020, INSTICC, 2020, p. 23.

[4] M.H.C. Garcia, A. Molina-Galan, M. Boban, J. Gozalvez, B. Coll-Perales, T. Şahin, A. Kousaridas, A tutorial on 5G NR V2X communications, IEEE Commun. Surv. Tutor. 23 (3) (2021) 1972–2026, http://dx.doi.org/10.1109/COMST.2021. 3057017.

[5] F. Hawlader, F. Robinet, R. Frank, Poster: Lightweight features sharing for real-time object detection in cooperative driving, in: 2023 IEEE Vehicular Networking Conference (VNC), 2023, pp. 159–160, http://dx.doi.org/10.1109/VNC57357. 2023.10136339.

[6] Z. Pan, J. Lei, Y. Zhang, X. Sun, S. Kwong, Fast motion estimation based on content property for low-complexity h. 265/HEVC encoder, IEEE Trans. Broadcast. 62 (3) (2016) 675–684.

[7] Y. Matsubara, M. Levorato, Neural compression and filtering for edge-assisted real-time object detection in challenged networks, in: 2020 25th International Conference on Pattern Recognition (ICPR), IEEE, 2021.

[8] B. Deguerre, C. Chatelain, G. Gasso, Fast object detection in compressed jpeg images, in: 2019 IEEE Intelligent Transportation Systems Conference (ITSC), IEEE, 2019, pp. 333–338.

[9] S.A. Abdel Hakeem, A.A. Hady, H. Kim, 5G-V2X: Standardization, architecture, use cases, network-slicing, and edge-computing, Wirel. Netw. 26 (8) (2020) 6015–6041.

[10] J.B. Jocher Glenn, Ayush Chaurasia, A. Stoken, YOLOv5 (2020), 2022, https://github.com/ultralytics/yolov5 (Accessed 10 July 2022).

[11] A. Dosovitskiy, G. Ros, V. Koltun, CARLA: An open urban driving simulator, in: Conference on Robot Learning, PMLR, 2017.

[12] H. Xu, S. Liu, G. Wang, G. Liu, B. Zeng, Omnet: Learning overlapping mask for partial-to-partial point cloud registration, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021.

[13] C.-M. Huang, C.-F. Lai, The mobile edge computing (MEC)-based vehicle to infrastructure (V2I) data offloading from cellular network to VANET using the delay-constrained computing scheme, in: 2020 International Computer Symposium (ICS), IEEE, 2020.

[14] A. Islam, A. Debnath, M. Ghose, S. Chakraborty, A survey on task offloading in multi-access edge computing, J. Syst. Archit. 118 (2021).

[15] M.-a.E. Computing, Framework and reference architecture, ETSI GS MEC 003, rev. 2.2. 1, dec. 2020, 2020.

[16] M.A. Khan, E. Baccour, Z. Chkirbene, A. Erbad, R. Hamila, M. Hamdi, M. Gabbouj, A survey on mobile edge computing for video streaming: Opportunities and challenges, IEEE Access (2022).

[17] Y. Siriwardhana, P. Porambage, M. Liyanage, M. Ylianttila, A survey on mobile augmented reality with 5G mobile edge computing: architectures, applications, and technical aspects, IEEE Commun. Surv. Tutor. 23 (2) (2021) 1160–1192.

[18] S.-L.C. Tsao, Enhanced GTP: an efficient packet tunneling protocol for general packet radio service, in: ICC 2001. IEEE International Conference on Communications. Conference Record (Cat. No. 01CH37240), Vol. 9, IEEE, 2001, pp. 2819–2823.

[19] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, in: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, 2001, http://dx.doi.org/10.1109/CVPR.2001.990517.

[20] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), Vol. 1, 2005, pp. 886–893, http://dx.doi.org/10.1109/CVPR.2005.177.

[21] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014, http://dx.doi.org/10.1109/CVPR.2014.81.

[22] R. Girshick, Fast R-CNN, in: 2015 IEEE International Conference on Computer Vision (ICCV), 2015, http://dx.doi.org/10.1109/ICCV.2015.169.

[23] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: Towards real-time object detection with region proposal networks, IEEE Trans. Pattern Anal. Mach. Intell. (2017) http://dx.doi.org/10.1109/TPAMI.2016.2577031.

[24] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A.C. Berg, SSD: Single shot MultiBox detector, in: B. Leibe, J. Matas, N. Sebe, M. Welling (Eds.), Computer Vision – ECCV 2016, Springer International Publishing, Cham, 2016, pp. 21–37.

[25] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[26] M. Ehrlich, L. Davis, S.-N. Lim, A. Shrivastava, Analyzing and mitigating JPEG compression defects in deep learning, in: 2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW), 2021.

[27] S. Dodge, L. Karam, Understanding how image quality affects deep neural networks, in: 2016 Eighth International Conference on Quality of Multimedia Experience (QoMEX), 2016, http://dx.doi.org/10.1109/QoMEX.2016.7498955.

[28] R. Yu, D. Yang, H. Zhang, Edge-assisted collaborative perception in autonomous driving: A reflection on communication design, in: 2021 IEEE/ACM Symposium on Edge Computing (SEC), IEEE, 2021.

[29] H. Vanholder, Efficient inference with tensorrt, in: GPU Technology Conference, Vol. 1, 2016, p. 2.

[30] G.A. Kovács, L. Bokor, Integrating artery and Simu5G: A mobile edge computing use case for collective perception-based V2X safety applications, in: 2022 45th International Conference on Telecommunications and Signal Processing (TSP), IEEE, 2022, pp. 360–366.

[31] J. Li, H. Gao, T. Lv, Y. Lu, Deep reinforcement learning based computation offloading and resource allocation for MEC, in: 2018 IEEE Wireless Communications and Networking Conference (WCNC), IEEE, 2018.

[32] A. Belogaev, A. Elokhin, A. Krasilov, E. Khorov, I.F. Akyildiz, Cost-effective V2X task offloading in MEC-assisted intelligent transportation systems, IEEE Access 8 (2020).

[33] E.E. Marvasti, A. Raftari, A.E. Marvasti, Y.P. Fallah, R. Guo, H. Lu, Feature sharing and integration for cooperative cognition and perception with volumetric sensors, 2020, Preprint arXiv:2011.08317.

[34] E. Ye, P. Spiegel, M. Althoff, Cooperative raw sensor data fusion for ground truth generation in autonomous driving, in: IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), IEEE.

[35] F. Hawlader, R. Frank, Towards a framework to evaluate cooperative perception for connected vehicles, in: 2021 IEEE Vehicular Networking Conference (VNC), 2021, pp. 36–39, http://dx.doi.org/10.1109/VNC52810.2021.9644667.

[36] J. Ren, Y. Guo, D. Zhang, Q. Liu, Y. Zhang, Distributed and efficient object detection in edge computing: Challenges and solutions, IEEE Netw. 32 (6) (2018).

[37] M. Ahmed, S. Raza, M.A. Mirza, A. Aziz, M.A. Khan, W.U. Khan, J. Li, Z. Han, A survey on vehicular task offloading: Classification, issues, and challenges, J. King Saud Univ.-Comput. Inf. Sci. (2022).

[38] A. Ndikumana, K.K. Nguyen, M. Cheriet, Age of processing-based data offloading for autonomous vehicles in MultiRATs open RAN, IEEE Trans. Intell. Transp. Syst. (2022).

[39] G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, Y. Kwon, J. Fang, K. Michael, D. Montes, J. Nadar, P. Skalski, et al., Ultralytics/yolov5: v6. 1-tensorrt, tensorflow edge tpu and openvino export and inference, Zenodo (2022).

[40] W. Zhao, S. Liu, X. Li, X. Han, H. Yang, Fast and accurate wheat grain quality detection based on improved YOLOv5, Comput. Electron. Agric. 202 (2022) 107426.

[41] A. Varga, R. Hornig, An overview of the OMNeT++ simulation environment, in: 1st International ICST Conference on Simulation Tools and Techniques for Communications, Networks and Systems, 2010.

[42] M. Farasat, D.N. Thalakotuna, Z. Hu, Y. Yang, A review on 5G sub-6 GHz base station antenna design challenges, Electronics (2021).

[43] G. Nardini, D. Sabella, G. Stea, P. Thakkar, A. Virdis, Simu5G–An OMNeT++ library for end-to-end performance evaluation of 5G networks, IEEE Access (2020) http://dx.doi.org/10.1109/ACCESS.2020.3028550.

[44] A. Filali, A. Abouaomar, S. Cherkaoui, A. Kobbane, M. Guizani, Multi-access edge computing: A survey, IEEE Access 8 (2020).

[45] J. Winick, S. Jamin, Inet-3.0: Internet topology generator, Tech. rep., Technical Report CSE-TR-456-02, University of Michigan, 2002.

[46] Z. Ali, S. Lagén, L. Giupponi, R. Rouil, 3GPP NR V2X mode 2: overview, models and system-level evaluation, IEEE Access (2021).

[47] R. Frank, F. Hawlader, Poster: Commercial 5G performance: A V2X experiment, in: 2021 IEEE Vehicular Networking Conference (VNC), IEEE, 2021, pp. 129–130.

[48] T.S. Rappaport, Y. Xing, G.R. MacCartney, A.F. Molisch, E. Mellios, J. Zhang, Overview of millimeter wave communications for fifth-generation (5G) wireless networks—With a focus on propagation models, IEEE Trans. Antennas Propag. 65 (12) (2017).

[49] J. Liu, Q. Zhang, To improve service reliability for AI-powered time-critical services using imperfect transmission in MEC: An experimental study, IEEE Internet Things J. 7 (10) (2020) 9357–9371.

[50] FFmpeg, 2023 Available:, https://github.com/FFmpeg/FFmpeg.

[51] M. Uhrina, J. Bienik, M. Vaculik, Coding efficiency of hevc/h. 265 and vp9 compression standards for high resolutions, in: 2016 26th International Conference Radioelektronika (RADIOELEKTRONIKA), IEEE, 2016, pp. 419–423.

[52] X. Ma, High-resolution image compression algorithms in remote sensing imaging, Displays (2023) 102462.

[53] J. Gutierrez-Aguado, R. Peña-Ortiz, M. García-Pineda, J.M. Claver, Cloud-based elastic architecture for distributed video encoding: Evaluating H. 265, VP9, and AV1, J. Netw. Comput. Appl. 171 (2020) 102782.