Vehicle-to-Infrastructure Communication for Real-Time Object Detection in Autonomous Driving

Faisal Hawlader, François Robinet and Raphaël Frank
Interdisciplinary Centre for Security, Reliability and Trust (SnT)
University of Luxembourg, 29 Avenue J.F Kennedy, L-1855 Luxembourg
firstname.lastname@uni.lu

***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 have to be taken into account for real-time perception in autonomous vehicles. Larger object detection models tend to produce the best results, but are also slower at runtime. Since the most accurate detectors cannot 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 an object detection model and evaluate different offloading strategies. Using real hardware and network simulations, we compare different trade-offs between prediction quality and end-to-end delay. Since sending raw frames over the network implies additional transmission delays, we also explore the use of JPEG compression at varying qualities and measure its impact on prediction metrics. We show that models with adequate compression can be run in real-time on the cloud while outperforming local detection performance.**

***Index Terms*—5G; Cloud/Edge Computing; Perception; C-V2X; Autonomous Driving;**

I. 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 (i.e. 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 favour of real-time operation.

In this work, we focus on visual perception using a front-facing camera. The position and class of objects in the scene is needed to plan collision-free paths in autonomous driving and should be available in real-time. We follow the existing literature and aim to perform object detection at a rate of 20Hz [1]. We investigate different variants of the YOLOv5 [2] detection model, which offer different detection qualities at different inference times. As illustrated in Table I, larger models generally perform better. However, higher performance comes at the cost of increased computational requirements. Due to the limited computing resources and power available in the car, running larger models can prove difficult.

One alternative is to offload some computations where resources are available. Compute capabilities are less limited on Multi-access Edge Computing (MEC) platforms, and the best hardware is available in the cloud. Offloading some perception computation to the cloud can be appropriate in some situations. However, data offloading to MEC or cloud devices adds some additional transmission latency, which might not be acceptable for time-sensitive applications. With the promise of new 5G technologies supported by C-V2X, data offloading is an interesting option to complement local perception in autonomous driving.

In this work, we explore data offloading strategies for remote object detection on edge and cloud devices. We aim to evaluate these strategies on their detection quality, as well as their compliance with end-to-end latency requirements using a realistic communication channel model.

Our contributions are as follows:

- We create a synthetic dataset to train an object detection model and evaluate the proposed offloading strategies.
- We investigate the transfer of camera frames and their processing on edge or cloud platforms. Using real hardware and network simulations, we compare different trade-offs between prediction quality and end-to-end delay.
- Since sending raw frames over the network implies additional transmission delays, we also explore the use of JPEG compression at varying qualities and measure its impact on prediction metrics.

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

II. Related work

A. V2I communication

As explained in [3, 4] using vehicle-to-infrastructure (V2I) communications, autonomous cars generally offload sensor

 ISBN: 978-3-903176-56-0

data processing to a server. The server could be placed on the edge using the 5G MEC [5] or in a cloud with higher computational resources [6]. V2I communications are carried out using the upload / download path [7]. 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 [8]. Offloading sensor data to the cloud through V2I adds additional transmission latency [9]. According to [7], this communication latency could be reduced using MEC, which requires a low transmission delay compared to the cloud. However, edge devices are also subject to limited resources that could limit application needs.

B. Object detection & 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 [10], or the Histogram-of-Gradient detector [11]. 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 [12–14]. Although two-stage models achieve impressive accuracy, their computational complexity makes real-time operation a challenge. To remedy this, the SSD [15] and YOLO [16] and family of models propose to combine 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 [2], 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 JPEG compression on detection performance, for varying compression qualities. Existing work has shown that JPEG compression negatively affects the performance of models trained on uncompressed frames [17, 18]. In the case of object detection, this effect is particularly noticeable at lower compression qualities. Performances deteriorate rapidly for moderate to heavy compression [17].

C. Perception using C-V2X communication

With the new features of 5G technologies, including edge and cloud computing, C-V2X technology has become more widespread [1, 5, 8]. C-V2X appears qualified to support advanced applications [19, 20], such as collective perception [21]. Processing of perception sensor data using the on board car computer might not always be an option due to latency constraints [6]. However, collective perception using the V2X service allows cars to potentially offload sensory data to edge and cloud for resource-intensive computations [4, 22]. 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 [4, 6–8]. However, the perception of surrounding objects requires real-time detection that is fast processing and low latency and cannot be arbitrarily obtained, as it is highly dependent on where we process the data. In [21, 23, 24], 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 emphasises data communication to reduce transmission overhead without evaluating the impact on perception quality. In principle, offloading raw sensor data is excellent for perception accuracy [25], but could increase transmission costs [26]. However, offloading compressed data can save network resources, but it might degrade detection quality [27]. The literature does not yet provide a trade-off between detection quality and end-to-end processing.

	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

TABLE I: Average Precision results for different model variants (AP@50). Model variants are detailed in Section III-A

III. Methodology

This section describes our initial efforts toward estimating the end-to-end delay of real-time object detection. Based on previous work, our objective is to perform object detection at a rate of 20Hz [1] without degrading the detection quality. In this context, end-to-end delay heavily depends on where we perform the computation. If the car chooses to offload the data to an edge or cloud, the car must exchange the raw sensor data, which V2X technologies may not support due to bandwidth needs [26].

A. 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 obtainable. However, reaching high detection quality and decreasing the inference delay remain a challenge [28]. 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 II, where local represents the on board device of an autonomous car that has limited processing power. However, the compute capabilities are less constrained on edge platforms, and the very best hardware is available in the cloud. This choice is made based on the research in [8, 29]. To perform object detection, we use YOLOv5 [2] as it has been demonstrated to have superior performance (that is, accuracy

Platform	Scenario / Model	Hardware configuration
Local (≈20W)	YOLOv5 small 157 layers, 7M params 640x640 Resolution	NVIDIA Jetson Xavier NX SoC Volta GPU, 384 CUDA cores Carmel ARMv8.2 CPU@1.9GHz
Edge (≈100W)	YOLOv5 large 267 layers, 46M params 640x640 Resolution	Laptop with GeForce GTX 1650 Turing GPU, 896 CUDA cores Intel i9-9980HK @2.4GHz
Cloud (≈450W)	YOLOv5 large high-res 346 layers, 76M params 1280x1280 Resolution	HPC node with Tesla V100 Volta GPU, 5120 CUDA cores Intel Xeon G6132 @2.6 Ghz

TABLE II: Based on inference time constraints, we selected distinct platforms to run three models of different sizes. Fig. 1 compares inference times for different models and platforms.

and latency). YOLOv5 offers various model sizes ranging from small to large. We observe that larger models are beneficial considering the detection quality as reported in Table I, but there are inference timing constraints that must be taken into account. In this context, we want the best possible model for each platform that fits 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 50ms and as such is not suited to run on the local platform. In view of the 20Hz detection speed, we decided to investigate the use of the small model on the local, the large on the edge, and the large high-res one on the cloud. The exact YOLOv5 version with the corresponding resolution is summarised in TableII.

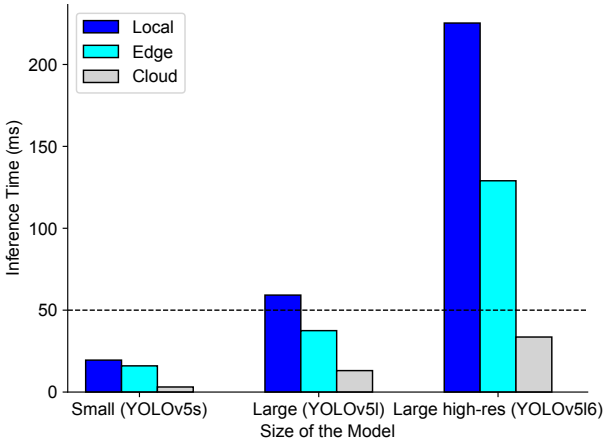


Fig. 1: Model inference time comparison between different model sizes on different platforms. We use half-precision floating-point computation at inference time in order to speed up computation.

B. 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

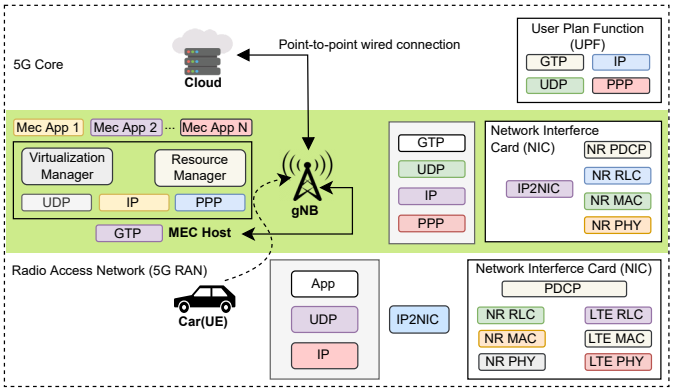


Fig. 2: Architecture of the end-to-end network simulation framework, showing the main elements of the 5G radio access network (RAN), including the multi-access edge computing (MEC) host-level components with a User Equipment (UE).

Simu5G [7] which is a OMNeT++ based discrete event network simulation library [30]. We focus on two scenarios, namely, perception data offloading with MEC and perception data offloading with cloud considering C-V2X technologies. Furthermore, we describe the main components of Simu5G used to model the scenarios considered. The network environment we consider consists of a RAN and a 5G Core Network (CN). The RAN has a single 5G base station (BS), which is an expansion of 4G base stations known as eNB [31]. One user equipment (UE) is attached to the BS, which is this use case is a vehicle. We also placed a MEC host close (500m) to the BS connected to a wired network, so MEC can interact closely with BS and obtain fast information from the RAN user. The gNB is then connected to a cloud server via CN. The cloud is located 1000km [6], away from the gNB base station. The different components of the framework are shown in Fig. 2.

5G Core Network (CN): In this work, we consider a standalone version of the 5G core network [32], which meets our requirements and is also available for simulation using Simu5G. The core network consists of a user plane function (UPF) that supports wired connection between the RNA and the cloud. A Point-to-Point (PPP) network interface is used to connect eNB to the cloud through a wired connection [33]. The GPRS [9] tunnelling protocol (GTP) is used to route IP datagrams (UDP) and establish a communication channel between gNB and the cloud.

Multi-access Edge Computing (MEC): The applications of MEC are growing and several standardisation initiatives are being carried out to provide a successful integration of MEC into the 5G network [34]. This work considers a simplified MEC host-level architecture in accordance with the European Telecommunications Standards Institute (ETSI) reference [5]. 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. We placed the MEC 500m away from the gNB [7]. A PPP wired connection with 100G data rate is used to connect the MEC to gNB [35].

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 [7].

User equipment (UE): As defined in the ETSI [5] and 3GPP specifications [36], 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 [6] and 5G NR [20], as shown in Fig. 2.

Parameter Name	**Value**
Carrier frequency	3.6 GHz [37]
gNB Tx Power	46 dBm
Path loss model	[7], Urban Macro (UMa)
Fading + Shadowing model	Enable, Long-normal distribution
Number of repetitions	200
Path loss model	3GPP-TR 36.873 [38]
UDP Packet size	4096 B [39]
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

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

C. Perception Aspects

Dataset generation: In order to train the YOLOv5 models described in Table II, we built a synthetic dataset using the CARLA simulator [40]. CARLA allows us to simulate a camera-equipped car driving in a rendered town environment, and to access 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 every second 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 data is split into three subsets: 6000 frames are used for training, 2000 for validation, and we reserve the remaining 2000 frames for testing.

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.

IV. Results

This section describes the experiments carried out to evaluate our proposed data offloading strategies. We perform experiments on the hardware described in Section II, and we analyse both the end-to-end delay and the detection quality.

A. End-to-end delay evaluation

To be able to compare the performance/latency trade-off of the three scenarios described in Section III-A and Table II, 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 local 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. We obtain a local end-to-end delay of 19.5 ms.

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 summarised in Table III. In these situations, the end-to-end delay includes compression, transmission, decompression and inference. The time to send the results back is not shown, but 0.43 ms (see Table III) extra latency is added to the transmission delay, assuming that the raw detection results will fit into a single packet. Sending raw uncompressed frames to remote devices results in large transmission delays because of the size of the data. We measured an average transmission time 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 a higher resolution frame being processed by the cloud model.

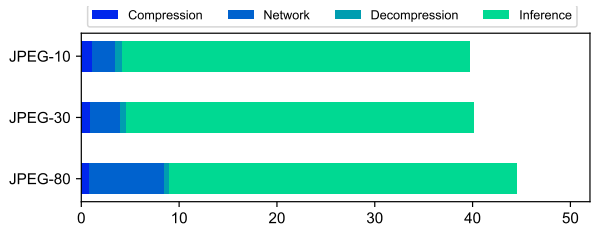
Since these delays are not acceptable in most practical perception scenarios, we investigate the use of JPEG compression to reduce them. For fast compression and decompression, we rely on the libjpeg-turbo library [41]. Compression always

occurs on the local device, and decompression happens on either the edge or cloud device depending on the scenario. As illustrated in Table IV, JPEG compression can drastically reduce the size of the frame to be transmitted, allowing real-time remote object detection when C-V2X is available.

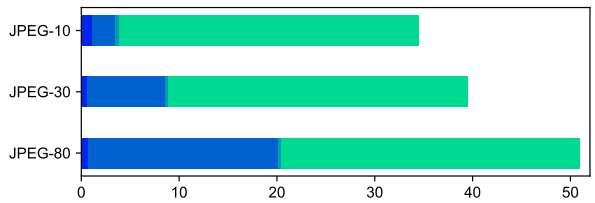
Platform	JPEG Quality	Avg. data size (% of original)	End-to-end delay (ms)
Edge 640×640	No compression	1.23 MB (100%)	123.2
	JPEG-80	63.7 KB (5.2%)	44.54
	JPEG-30	27.2 KB (2.2%)	40.13
	JPEG-10	14.6 KB (1.2%)	39.68
Cloud 1280×1280	No compression	4.92 MB (100%)	521.7
	JPEG-80	185.7 KB (3.8%)	50.9
	JPEG-30	81.3 KB (1.7%)	39.52
	JPEG-10	45.9 KB (0.9%)	34.44

TABLE IV: Amount of data to transfer for the edge and cloud scenarios, at different JPEG compression qualities. Last column refers to end-to-end delay in vehicle-to-edge/cloud scenario.

A breakdown of average end-to-end delays for different compression qualities is displayed in Fig. 3. 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 times are constant for a given platform since 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 20Hz on both platforms. The next section will investigate how compression impacts detection quality.



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



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

Fig. 3: Breakdown of average end-to-end delay (ms) into compression overhead, network transfer, and inference times.

B. 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, 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 JPEG compression. However, excessive compression affects detection quality, degrading the mAP. This section aims to determine the best trade-offs between detection quality and end-to-end delay. These trade-offs are illustrated in Fig. 4.

As expected, local operation is the fastest in terms of latency, and obtains 64% mAP. As we can see from Fig. 4, the combination of hardware and model of the edge platform either leads to worse results in the JPEG-30 case (62% mAP) and slightly better than Local in the JPEG-80 case (67% mAP). The end-to-end delays are 40.13 ms and 44.54 ms respectively. The Edge scenario examined here is therefore 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 since it offers better performance with both JPEG-30 and JPEG-80, with 69% and 80% mAP respectively. Meanwhile, the end-to-end delays are kept under 50ms, respecting the 20Hz 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 the offloading harmful.

We also evaluate the detection quality separately for the three classes of interest: pedestrians, vehicles and traffic lights. The purpose of this evaluation is to understand the impact of compression on different classes. The results are depicted in Table V. We observe that compression has a disproportionate impact on AP for classes that are typically smaller in scale. Indeed, when enabling JPEG-80 compression, AP decreases by -16% for pedestrians and by -4% for traffic lights, while vehicles see a $+2\%$ AP increase. This is not surprising, since JPEG compression artifacts will impact smaller objects more than larger ones. Pedestrians are rarer in the dataset compared to traffic lights, and their appearance also varies much more. These conditions are challenging for object detectors.

V. 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

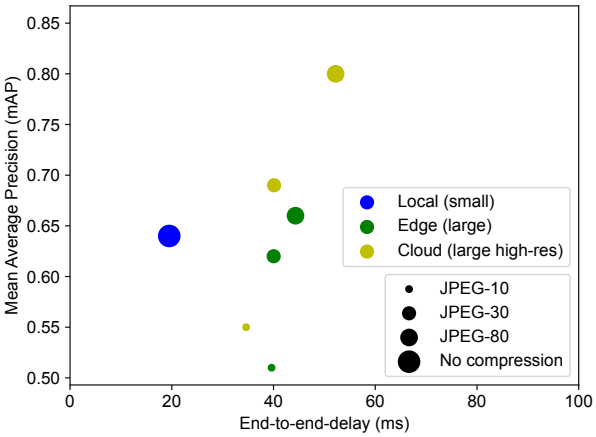


Fig. 4: Trade-off between mean average precision (mAP) and end-to-end delay for different platforms and compression qualities. The end-to-end delay correspond to the total of compression, network, decompression, and object detection inference delays.

Compression quality	Platform	Pedestrian	Vehicle	Traffic light
No compression	Local	0.30	0.79	0.80
	Edge	0.36	0.81	0.82
	Cloud	0.81	0.86	0.89
JPEG-80	Edge	0.41	0.83	0.78
	Cloud	0.65	0.88	0.85
JPEG-30	Edge	0.35	0.77	0.74
	Cloud	0.43	0.84	0.81
JPEG-10	Edge	0.23	0.73	0.55
	Cloud	0.24	0.78	0.62

TABLE V: Per-class AP for different compression qualities and platforms. The input resolutions are 640×640 for the local and edge platforms, and 1280×1280 for the cloud model.

offload object detection to edge or cloud devices using C-V2X. We have compared these strategies in terms of their detection quality 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 analysed how the use of heavy JPEG compression can reduce the frame size by up to 98% to enable real-time remote processing. Our experimental results show that excessive compression affects the detection quality compared to raw frames, particularly for the pedestrian class. We show that models with adequate compression can be run in real-time on the cloud while outperforming local detection performance.

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.

Acknowledgments

This work is supported by the Fonds National de la Recherche of Luxembourg (FNR), under AFR grant agreement No 17020780 and project acronym *ACDC*.

References

[1] S. A. Abdel Hakeem, A. A. Hady, and H. Kim, "5g-v2x: Standardization, architecture, use cases, network-slicing, and edge-computing," *Wireless Networks*, vol. 26, no. 8, pp. 6015–6041, 2020.

[2] J. B. Jocher Glenn, Ayush Chaurasia and A. Stoken, "Yolov5 (2020)," *https://github.com/ultralytics/yolov5.*, Accessed 10 July 2022.

[3] C.-M. Huang and 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, pp. 1–6.

[4] A. Islam, A. Debnath, M. Ghose, and S. Chakraborty, "A survey on task offloading in multi-access edge computing," *Journal of Systems Architecture*, vol. 118, p. 102225, 2021.

[5] M.-a. E. Computing, "Framework and reference architecture, etsi gs mec 003, rev. 2.2. 1, dec. 2020."

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

[7] G. Nardini, G. Stea, A. Virdis, and D. Sabella, "Simu5g: a system-level simulator for 5g networks," in *SIMULTECH 2020*. INSTICC, 2020.

[8] Y. Siriwardhana, P. Porambage, M. Liyanage, and M. Ylianttila, "A survey on mobile augmented reality with 5g mobile edge computing: architectures, applications, and technical aspects," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1160–1192, 2021.

[9] 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.

[10] P. Viola and 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*, vol. 1, 2001, pp. I–I.

[11] N. Dalal and 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 vol. 1.

[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587.

[13] R. Girshick, "Fast r-cnn," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448.

[14] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

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

[16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[17] M. Ehrlich, L. Davis, S.-N. Lim, and A. Shrivastava, "Analyzing and mitigating jpeg compression defects in deep learning," in *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2021, pp. 2357–2367.

[18] S. Dodge and L. Karam, "Understanding how image quality affects deep neural networks," in *2016 Eighth International Conference on Quality of Multimedia Experience (QoMEX)*, 2016, pp. 1–6.

[19] R. Yu, D. Yang, and 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, pp. 371–375.

[20] H. Vanholder, "Efficient inference with tensorrt," in *GPU Technology Conference*, vol. 1, 2016, p. 2.

[21] G. A. Kovács and 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.

[22] J. Li, H. Gao, T. Lv, and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for mec," in *2018 IEEE Wireless communications and networking conference (WCNC)*. IEEE, 2018, pp. 1–6.

[23] A. Belogaev, A. Elokhin, A. Krasilov, E. Khorov, and I. F. Akyildiz, "Cost-effective v2x task offloading in mec-assisted intelligent transportation systems," *IEEE access*, vol. 8, pp. 169 010–169 023, 2020.

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

[25] E. Ye, P. Spiegel, and M. Althoff, "Cooperative raw sensor data fusion for ground truth generation in autonomous driving," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020, pp. 1–7.

[26] F. Hawlader and R. Frank, "Towards a framework to evaluate cooperative perception for connected vehicles," in *2021 IEEE Vehicular Networking Conference (VNC)*, 2021, pp. 36–39.

[27] J. Ren, Y. Guo, D. Zhang, Q. Liu, and Y. Zhang, "Distributed and efficient object detection in edge computing: Challenges and solutions," *IEEE Network*, vol. 32, no. 6, pp. 137–143, 2018.

[28] M. Ahmed, S. Raza, M. A. Mirza, A. Aziz, M. A. Khan, W. U. Khan, J. Li, and Z. Han, "A survey on vehicular task offloading: Classification, issues, and challenges," *Journal of King Saud University-Computer and Information Sciences*, 2022.

[29] A. Ndikumana, K. K. Nguyen, and M. Cheriet, "Age of processing-based data offloading for autonomous vehicles in multirats open ran," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[30] A. Varga and 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.

[31] M. Farasat, D. N. Thalakotuna, Z. Hu, and Y. Yang, "A review on 5g sub-6 ghz base station antenna design challenges," *Electronics*, vol. 10, no. 16, p. 2000, 2021.

[32] G. Nardini, D. Sabella, G. Stea, P. Thakkar, and A. Virdis, "Simu5g–an omnet++ library for end-to-end performance evaluation of 5g networks," *IEEE Access*, vol. 8, pp. 181 176–181 191, 2020.

[33] H. Xu, S. Liu, G. Wang, G. Liu, and 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, pp. 3132–3141.

[34] A. Filali, A. Abouaomar, S. Cherkaoui, A. Kobbane, and M. Guizani, "Multi-access edge computing: A survey," *IEEE Access*, vol. 8, pp. 197 017–197 046, 2020.

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

[36] Z. Ali, S. Lagén, L. Giupponi, and R. Rouil, "3gpp nr v2x mode 2: overview, models and system-level evaluation," *IEEE Access*, 2021.

[37] R. Frank and F. Hawlader, "Poster: Commercial 5g performance: A v2x experiment," in *2021 IEEE Vehicular Networking Conference (VNC)*. IEEE, 2021, pp. 129–130.

[38] T. S. Rappaport, Y. Xing, G. R. MacCartney, A. F. Molisch, E. Mellios, and J. Zhang, "Overview of millimeter wave communications for fifth-generation (5g) wireless networks—with a focus on propagation models," *IEEE Transactions on antennas and propagation*, vol. 65, no. 12, pp. 6213–6230, 2017.

[39] J. Liu and Q. Zhang, "To improve service reliability for ai-powered time-critical services using imperfect transmission in mec: An experimental study," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9357–9371, 2020.

[40] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[41] P. repository, "Libjpeg-turbo," *https://github.com/libjpeg-turbo/libjpeg-turbo*, Accessed 01 November 2022.