2025 17th International Conference on COMmunication Systems and NETworks (COMSNETS) | 979-8-3315-3119-5/25/$31.00 ©2025 IEEE | DOI: 10.1109/COMSNETS63942.2025.10885768

Cloud-Assisted Autonomous Driving over Wireless Network

Najiya Naj*, Debopam Bhattacherjee†, Arani Bhattacharya*
{najiyan, arani}@iiitd.ac.in, debopamb@microsoft.com
* IIIT-Delhi, † Microsoft Research India

***Abstract*—Autonomous driving systems utilize machine learning models to identify the right strategy to control vehicles. Since these systems are safety-critical, high accuracy, especially in complex scenarios, is increasingly ensured by designing larger machine learning (ML) models that require higher execution time and memory. Such larger ML models are challenging to process on onboard systems, as specialized processors with large memories tend to be expensive. One possibility of utilizing such large ML models is to leverage cloud computing in real time. However, cloud services can only be accessed using the wireless network, which can vary depending on the location and network congestion. In this work, we design a system that adapts to such dynamic network conditions, where the amount of data sent depends on the available network bandwidth. This adaptation technique (i) tunes the quality of the sensor data and (ii) depends more on the LIDAR data for local processing when the network fails. Our system caches additional data, such as HD maps, from the cloud whenever it predicts that the network bandwidth is about to reduce. Our preliminary results using an existing autonomous driving simulator called Carla shows that our technique is effective in practice.**

***Index Terms*—Autonomous Driving, Simulator, Object detection, Heavyweight model, adaptive encoding.**

I. Introduction

While autonomous driving systems have significantly advanced, their promise of full autonomy has not been achieved yet. For example, The key reason is the presence of challenging edge cases, especially ones not commonly observed during training of the machine learning models [9], [23]. The standard strategy of improving the accuracy has been to design larger and more compute-intensive models [16]. Running such models requires specialized processors called graphical processing units (GPUs) with larger memories. However, such GPUs with large memories tend to be significantly more expensive, as shown in Fig. 1. Thus, a safer and more efficient strategy for enabling autonomous driving is needed.

To enable such safer and more efficient autonomous driving, utilization of teleoperation or cloud-assistance has been proposed by a number of recent studies [24], [25], [21]. Teleoperation allows human operators to intervene in critical situations, aiding in navigation through complex environments. On the other hand, cloud-assisted architectures address local processing limitations by offloading compute-heavy tasks to remote servers, which provide scalable computational power [20]. This approach enables the use of large-scale models and data sharing across vehicles for enhanced situational awareness. Additionally, continuous analysis of real-time and historical data helps the system manage edge cases and reduces the processing load on onboard systems.

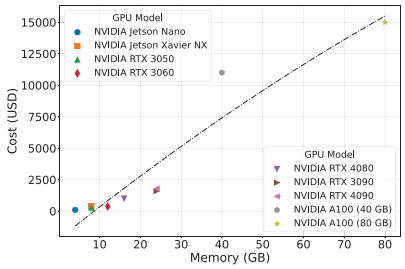


Fig. 1: GPU memory vs. monetary cost comparison. We note that there is almost a linear increase in the cost of GPU memory from 4GB to 80GB.

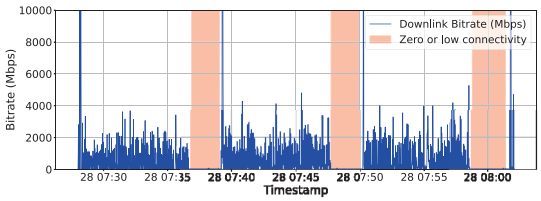


Fig. 2: Bitrate fluctuations over time taken obtained from a public dataset in Ireland while driving. We note that there are periods of almost no connectivity.

Such remote-assistance solutions, however, face challenges such as network latency and connectivity issues [13], [22], which can cause delays and impact safety in critical situations. A hybrid approach, combining both local and cloud resources, offers a promising solution to autonomous vehicle [6]. Time-sensitive tasks, like obstacle detection, are processed locally for faster responses, while more complex tasks are offloaded to the cloud when network conditions allow. The key challenge of using such a hybrid approach is to adapt the amount of offloading and vehicle's decisions based on the conditions of the network.

For example, when a vehicle travels at high speeds, such as on highways, network connectivity can often degrade due to high distance from cell towers or interference from physical structures like tunnels. These areas often present poor or no connectivity, making real-time data transmission unreliable. A vehicle in such a situation may experience delays in receiving crucial updates, which could compromise its ability

to detect obstacles or adapt to rapidly changing environments.

To resolve this challenge, our system uses lightweight models on the local machine for critical tasks like obstacle detection. These models run with minimal computation, ensuring fast decision-making even when connectivity is weak. When the vehicle exits a tunnel or slows down, network conditions improve, enabling offloading more complex tasks—such as real-time mapping, high-definition object recognition, or detailed prediction models—to the cloud. Additionally, the local machine can cache data during periods of poor connectivity, allowing the vehicle to continue operating autonomously using previously stored information until connectivity is restored. We integrate this hybrid approach into an existing simulator of autonomous vehicles called Pylot. Our system's hybrid approach ensures continuous operation, minimizes risk in low-connectivity areas, and efficiently leverages cloud resources when available, offering a scalable solution for autonomous driving in diverse environments.

II. Background & Motivation

We now discuss the working of the simulator Pylot [7] as we use it to build our hybrid approach. Pylot decomposes the process of decision-making into multiple layers, namely perception, prediction, planning and control. The perception module focuses on understanding the current situation on the road, using techniques such as object detection, tracking and segmentation for identifying lanes. The prediction module uses algorithms to identify the position of the objects in the near future, using techniques such as linear extrapolation. The planning module identifies the right strategy for controlling the vehicle, whereas the control module decides the right action to execute the decisions taken by the planning module. Pylot integrates multiple ML models and algorithms for each of these techniques in a modular fashion. It is integrated with a vehicle simulator called CARLA [3] that can be used to run a vehicle in different traffic scenarios. This makes it easy to experiment using different ML models and algorithms.

A significant challenge of using Pylot is that it typically requires powerful GPUs within the vehicle to run heavyweight models in complex scenarios, which can be both cost-prohibitive and computationally demanding. Furthermore, autonomous driving applications rely on high-resolution sensors, like cameras and LiDARs, which increase GPU memory and processing demands due to real-time data processing [15]. Meeting these demands locally adds significant hardware costs and power consumption, making real-time operation difficult. Fig. 1 shows that as GPU memory requirements increase to meet these demands, costs rise sharply, creating a barrier to fully relying on local resources. This limitation highlights the need for a hybrid approach.

An alternative approach is to utilize remote resources, which could be either a human operator or cloud servers with larger computing power. However, relying entirely on network connectivity can present risks in scenarios where network stability fluctuates or fails entirely. For example, on highways or in rural areas with weak signals, a sudden network loss could prevent the vehicle from accessing essential data and decision-making capabilities. Fig. 2 shows how the network fluctuates in a vehicle using actual traces [18], making it challenging to depend on the cloud servers to control a vehicle.

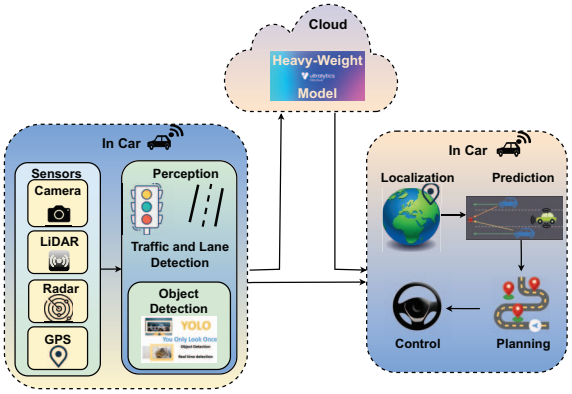


Fig. 3: Hybrid onboard-cloud architecture for autonomous driving

Thus, it is currently challenging to fully rely either on local processing or on remote resources. This challenge calls for an *adaptive* approach of deciding where to run the ML models, the type of ML models and algorithms to use, as well as controlling of vehicle based on the complexity of scenes as well as the network conditions. Furthermore, such an approach needs to be extensively tested using realistic simulations to ensure that it is deployable in practice.

III. Design & Methodology

Our system is built on top of Pylot and leverages both local processing and cloud computing to manage the demands of autonomous driving (Fig. 3). We first explain the design goals of the system:

1) **Use of lightweight models on vehicle:** The ML models and algorithms used on the vehicle should be small and lightweight enough to avoid the use of server-quality GPUs.

2) **Minimal adverse impact on vehicle's safety and speed:** The use of lightweight ML models and algorithms should have minimal adverse impact on the safety or speed of the autonomous vehicle. This is possible via opportunistic use of the cloud resources whenever network connectivity is sufficiently available.

3) **Safe operation of a vehicle under poor connectivity:** While cloud servers can be used, the vehicle should operate safely even during periods of poor connectivity.

We show our end-to-end architecture in Fig 3, which leverages both local processing and cloud computing to manage the demands of autonomous driving. Our approach is to duplicate the functionality of perception, prediction and planning on both the vehicle and the cloud server. The models and algorithms on the vehicles are lightweight, whereas

the ones on the cloud server are more compute-intensive[1]. We then design two distinct adaptation strategies. The first strategy is to intelligently switch between the lightweight and heavyweight models depending on the road condition and available network bandwidth. The second strategy is to identify when network conditions are likely to get worse and cache relevant data proactively on the vehicle to enable better decisions using lightweight models.

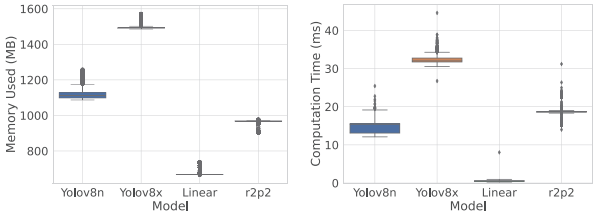
1) **Hybrid Model Deployment**: We classify tasks between local and cloud models to enable the use of lightweight models. Our system uses multiple models for perception, prediction, and planning, which run in parallel to handle diverse tasks. Some models, such as lane segmentation and trajectory planning, do not require real-time execution and involve higher computational demands, making them suitable for offloading to the cloud.

On the other hand, the local system runs lightweight models for tasks requiring immediate responses, such as near-field object detection and obstacle avoidance. For example, in Fig. 4a, we observe that Yolov8n has a significantly lower memory requirement than Yolov8x. Note that since multiple cameras are present on a single vehicle, this difference in memory requirement becomes significantly higher. Furthermore, there are two prediction algorithms integrated in Pylot – linear and r2p2 [19]. The linear predictor, while inaccurate, requires significantly lower memory and lower computation time than r2p2 and is thus suitable for local use. We envisage our system to use local computation whenever it provides sufficient level of accuracy, i.e. it encounters scenes with relatively low complexity. We leave the strategy of classifying scenes into simple and complex for future work.

2) **Network-Based Dynamic Switching**: Our system monitors network conditions in real-time using metrics like Received Signal Strength Indicator (RSSI) and latency. Tasks requiring heavyweight models, like lane segmentation or long-range object detection, have less stringent real-time requirements. Thus, they are offloaded to the cloud when the network is stable. Furthermore, since the camera and LIDAR data is large in nature, we utilize video encoding before sending them. To do so, we utilize an adaptive encoding technique similar to the strategy used in live videoconferencing. This includes adjusting the quantization parameter (QP) and resolution of the video. Then, the system continuously adapts the allocation of tasks between local and cloud processing, guaranteeing uninterrupted operation of critical tasks even under varying network conditions. When network connectivity weakens or becomes unstable, the system switches to lightweight models for local processing, focusing on immediate tasks like near-field object detection.

3) **Proactive Caching and HD Map Integration**: A key strategy of our system is to identify areas of poor connectivity and proactively cache the relevant data to support local operations while driving through these areas. The stored data includes HD maps with static features like lane boundaries and intersections. This cached data reduces the requirement of perception from multiple cameras. Furthermore, it also reduces the need for planning, as all the static features are known to the vehicle.

[1]Note that it is also possible for a human operator instead of more heavy models on the cloud. Our approach would work equally well in such a scenario.



(a) (b)

Fig. 4: Overview of images: (a) Memory usage, (b) Computation time of lightweight vs. heavyweight models

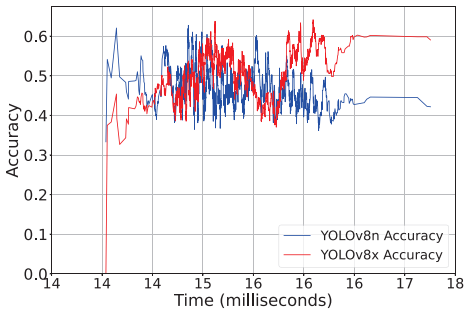


Fig. 5: Accuracy comparison of YOLO models

IV. Implementation & Evaluation

We evaluate the proposed hybrid onboard cloud architecture using two distinct hardware configurations to measure its efficiency and performance across various metrics:

Local Machine: We use an Intel(R) Xeon(R) Silver 4310 CPU with an NVIDIA GeForce RTX 3060 GPU to emulate the vehicle's onboard processing capabilities. This configuration allows us to evaluate the lightweight, real-time execution and in-vehicle setup.

Remote Machine (Cloud): For cloud processing, we used an NVIDIA GeForce RTX 4070 GPU to evaluate more complex and high computation tasks remotely.

The transfer of actual data from the local machine to the vehicle required us to make changes to the Pylot system. Our first change is to include encoding of the camera data in the form of H.264 video streams using ffmpeg. This reduces the amount of data that needs to be sent by over 50x, making it easier to use the cloud. Our second change is to further integrate local inference using the LIDAR data.

We focus on key metrics, including memory usage, computation time, and model accuracy in perception and prediction tasks. Fig 4 shows the memory usage and computation time across different models of perception module that uses the

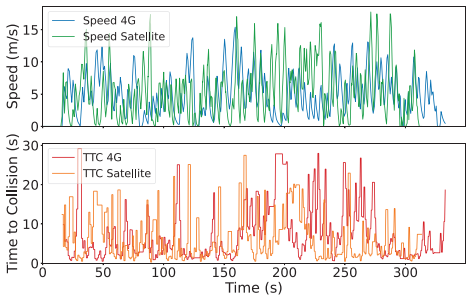


Fig. 6: Speed and time to collision. Both 4G and satellite network provides similar speeds and time-to-collision.

YOLO model for object detection, while the prediction module utilizes Linear and r2p2 models for trajectory estimation.

From Fig. 4a and 4b we observe that lightweight models, such as YOLOv8n and the Linear model, require significantly less memory and computation time. On the other hand, heavyweight models like YOLOv8x and r2p2 require more resources, making them suitable for cloud deployment in scenarios where computational capacity is sufficient.

In Fig. 5, we compare the accuracy of YOLO models. It shows that YOLOv8x achieves higher accuracy than the lightweight YOLOv8n model. This greater accuracy, along with its higher computation time, supports the use of YOLOv8x as a heavyweight model best for cloud deployment under favourable network conditions.

We also perform an experiment using CARLA to evaluate remote operation over 4G and satellite networks. We study the impact of the network on the quality of driving, measured using speed and time to collision, in Fig. 6. This experiment shows the speed and time-to-collision across a total of 5 iterations of teleoperated driving. We find that both the satellite traces and 4G network provide similar values of speed and time-to-collision, with the time-to-collision having median values of 5.28s and 4.36s, respectively, indicating that remote operation is feasible over both satellite and cellular networks.

V. Related Work

Since remote-assistance systems depend on real-time interactions between local machines, human operators or the cloud, they are particularly sensitive to delays and require robust strategies to manage delays [17]. For instance, Kettwich et al. [11] identified failure scenarios in teleoperated driving or cloud-assistance. Tools such as CARLA [3] allow simulation of urban driving environments, which is crucial to evaluate autonomous vehicle performance under various traffic conditions. An extension Scenario Runner [1] within CARLA allows users to create and test complex driving scenarios, including various traffic situations and environmental conditions. TELECARLA [10] builds upon these capabilities by making teleoperation research more accessible by the use of standard, off-the-shelf components. Our system leverages Carla and ScenarioRunner for its experiments but also integrates both lightweight models on vehicles and transfer of sensor data to remote server.

In real-world applications, maintaining stable and low-latency connectivity is essential. Solutions like DriveU.auto [4] and Halo [8] focus on teleoperation of vehicles. However, the reliance on cellular networks introduces potential variability in data transmission, with issues like signal strength fluctuations, network congestion, and environmental interference causing delays and affecting reliability [5]. To address these challenges, edge computing enables local sensor data processing to reduce network dependency, while V2X communication improves redundancy and reliability under variable conditions [14]. Moreover, offloading autonomous driving tasks to edge servers reduces latency and computational demands on vehicles, ensuring seamless real-time operation [2], Alternative methods have been investigated to address connectivity challenges. Laniewski et al. [12] examined the use of Starlink in mobile scenarios across Central Europe, discovering its potential for vehicular communication in areas where cellular coverage is inadequate. Their study, however, also noted difficulties in maintaining consistent connectivity during motion, especially in urban environments. The key goal of our system is to utilize these observations to design better adaptation techniques.

VI. Conclusion & Future Work

In this work, we present a hybrid onboard-cloud architecture for autonomous driving that combines local processing with cloud computing to optimize performance. Lightweight models run directly on the vehicle, reducing the need for high-performance hardware and maintaining safety and responsiveness. The system uses heavyweight models in the cloud to achieve higher accuracy when network connectivity is stable. It adapts dynamically to varying network conditions, ensuring continuous operation even during connectivity issues by leveraging cached data. Preliminary results confirm that this approach provides a reliable and efficient solution for real-time autonomous driving tasks.

For future work, we will focus on evaluating LiDAR-based object detection in different scenarios and compare performance across camera, LiDAR, and combined sensor configurations. We also plan to change the encoding parameters and develop a better understanding of which parameters are suitable under different circumstances. We also plan to study the impact of network conditions on the vehicle's control decisions, and the system will be further optimized for real-time decision-making in low-connectivity environments.

Acknowledgement

This work was supported by All India Council of Technical Education (AICTE) Doctoral Fellowship, awarded to Najiya Naj (Student ID: S2022405), to do research under the umbrella of smart cities.

REFERENCES

[1] CARLA Simulator. Scenario runner. https://github.com/carla-simulator/scenario_runner. Accessed: 2018-12-22.

[2] M. Cui, S. Zhong, B. Li, X. Chen, and K. Huang. Offloading autonomous driving services via edge computing. *IEEE Internet of Things Journal*, 7(10):10535–10547, 2020.

[3] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16, Mountain View, US, 2017. PMLR.

[4] DriveU.auto. DriveU.auto. https://driveu.auto/, 2023. Accessed: 2022-08-23.

[5] D. Garcia-Roger, E. E. González, D. Martín-Sacristán, and J. F. Monserrat. V2x support in 3gpp specifications: From 4g to 5g and beyond. *IEEE access*, 8:190946–190963, 2020.

[6] L. Gillam, K. Katsaros, M. Dianati, and A. Mouzakitis. Exploring edges for connected and autonomous driving. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 148–153. IEEE, 2018.

[7] I. Gog, S. Kalra, P. Schafhalter, M. A. Wright, J. E. Gonzalez, and I. Stoica. Pylot: A modular platform for exploring latency-accuracy tradeoffs in autonomous vehicles. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8806–8813. IEEE, 2021.

[8] Halo Teleoperation. halo.car. https://halo.car/, 2021. Accessed: 2022-08-23.

[9] A. J. Hawkins. Dude, where's my self-driving car? https://www.theverge.com/24065447/self-driving-car-autonomous-tesla-gm-baidu, 2024. Accessed: 2024-02-13.

[10] M. Hofbauer, C. B. Kuhn, G. Petrovic, and E. Steinbach. Telecarla: An open source extension of the carla simulator for teleoperated driving research using off-the-shelf components. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 335–340, Las Vegas, USA, 2020. IEEE.

[11] C. Kettwich, A. Schrank, H. Avsar, and M. Oehl. What if the automation fails?–a classification of scenarios in teleoperated driving. In *13th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, pages 92–96, NY, USA, 2021. AutomotiveUI'21 Adjunct.

[12] D. Laniewski, E. Lanter, S. Beainn, J. Dunker, M. Dückers, and N. Aschenbruck. Starlink on the road: A first look at mobile starlink performance in central europe. In *2024 8th Network Traffic Measurement and Analysis Conference (TMA)*, pages 1–8. IEEE, 2024.

[13] H. Lin, S. Zeadally, Z. Chen, H. Labiod, and L. Wang. A survey on computation offloading modeling for edge computing. *Journal of Network and Computer Applications*, 169:102781, 2020.

[14] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi. Edge computing for autonomous driving: Opportunities and challenges. *Proceedings of the IEEE*, 107(8):1697–1716, 2019.

[15] J. Mendez, M. Molina, N. Rodriguez, M. P. Cuellar, and D. P. Morales. Camera-lidar multi-level sensor fusion for target detection at the network edge. *Sensors*, 21(12):3992, 2021.

[16] K. Muhammad, A. Ullah, J. Lloret, J. Del Ser, and V. H. C. de Albuquerque. Deep learning for safe autonomous driving: Current challenges and future directions. *IEEE Transactions on Intelligent Transportation Systems*, 22(7):4316–4336, 2020.

[17] S. Neumeier, P. Wintersberger, A.-K. Frison, A. Becher, C. Facchi, and A. Riener. Teleoperation: The holy grail to solve problems of automated driving? sure, but latency matters. In *Proceedings of the 11th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, pages 186–197, Hawaii, USA, 2019. ITSC.

[18] D. Raca, D. Leahy, C. J. Sreenan, and J. J. Quinlan. Beyond throughput, the next generation: A 5g dataset with channel and context metrics. In *Proceedings of the 11th ACM multimedia systems conference*, pages 303–308, 2020.

[19] N. Rhinehart, K. M. Kitani, and P. Vernaza. R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 772–788, 2018.

[20] M. Sajjad, M. Irfan, K. Muhammad, J. Del Ser, J. Sanchez-Medina, S. Andreev, W. Ding, and J. W. Lee. An efficient and scalable simulation model for autonomous vehicles with economical hardware. *IEEE transactions on intelligent transportation systems*, 22(3):1718–1732, 2020.

[21] P. Schafhalter, S. Kalra, L. Xu, J. E. Gonzalez, and I. Stoica. Leveraging cloud computing to make autonomous vehicles safer. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5559–5566, Detroit, USA, 2023. IEEE.

[22] A. Shakarami, M. Ghobaei-Arani, M. Masdari, and M. Hosseinzadeh. A survey on the computation offloading approaches in mobile edge/cloud computing environment: a stochastic-based perspective. *Journal of Grid Computing*, 18(4):639–671, 2020.

[23] J. Tabor, S. Dai, V. Sreenivasan, and S. Banerjee. μcity: a miniatured autonomous vehicle testbed. In *Proceedings of the 17th ACM Workshop on Mobility in the Evolving Internet Architecture*, pages 25–30, 2022.

[24] F. Tener and J. Lanir. Driving from a distance: challenges and guidelines for autonomous vehicle teleoperation interfaces. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–13, LA, USA, 2022. CHI'22.

[25] Y. Trabelsi, O. Shabat, J. Lanir, O. Maksimov, and S. Kraus. Advice provision in teleoperation of autonomous vehicles. In *Proceedings of the 28th International Conference on Intelligent User Interfaces*, pages 750–761, 2023.