Soar: Design and Deployment of A Smart Roadside Infrastructure System for Autonomous Driving

Shuyao Shi*, Neiwen Ling*, Zhehao Jiang*, Xuan Huang*, Yuze He, Xiaoguang Zhao, Bufang Yang, Chen Bian, Jingfei Xia, Zhenyu Yan, Raymond W. Yeung, and Guoliang Xing†

The Chinese University of Hong Kong, Hong Kong SAR, China

ABSTRACT

Recently, *smart roadside infrastructure (SRI)* has demonstrated the potential of achieving fully autonomous driving systems. To explore the potential of infrastructure-assisted autonomous driving, this paper presents the design and deployment of *Soar*, the first end-to-end SRI system specifically designed to support autonomous driving systems. *Soar* consists of both software and hardware components carefully designed to overcome various system and physical challenges. *Soar* can leverage the existing operational infrastructure like street lampposts for a lower barrier of adoption. *Soar* adopts a new communication architecture that comprises a bi-directional multi-hop I2I network and a downlink I2V broadcast service, which are designed based on off-the-shelf 802.11ac interfaces in an integrated manner. *Soar* also features a hierarchical DL task management framework to achieve desirable load balancing among nodes and enable them to collaborate efficiently to run multiple data-intensive autonomous driving applications. We deployed a total of 18 *Soar* nodes on existing lampposts on campus, which have been operational for over two years. Our real-world evaluation shows that *Soar* can support a diverse set of autonomous driving applications and achieve desirable real-time performance and high communication reliability. Our findings and experiences in this work offer key insights into the development and deployment of next-generation smart roadside infrastructure and autonomous driving systems.

*Co-primary authors.
†Corresponding author.

ACM MobiCom '24, November 18–22, 2024, Washington D.C., DC, USA

ACM ISBN 979-8-4007-0489-5/24/09…$15.00
https://doi.org/10.1145/3636534.3649352

CCS CONCEPTS

• **Computer systems organization** → **Sensor networks**;
• **Information systems** → *Sensor networks.*

KEYWORDS

Smart Roadside Infrastructure, Infrastructure-Assisted Autonomous Driving, V2X, Edge Computing

ACM Reference Format:

Shuyao Shi*, Neiwen Ling*, Zhehao Jiang*, Xuan Huang*, Yuze He, Xiaoguang Zhao, Bufang Yang, Chen Bian, Jingfei Xia, Zhenyu Yan, Raymond W. Yeung, and Guoliang Xing†. 2024. *Soar*: Design and Deployment of A Smart Roadside Infrastructure System for Au-tonomous Driving. In *International Conference On Mobile Comput-ing And Networking (ACM MobiCom '24), September 30–October 4, 2024, Washington D.C., DC, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3636534.3649352

1 INTRODUCTION

Autonomous driving is envisioned to revolutionize our transportation system. However, current pilot commercial deployments have posed major concerns regarding the safety of existing autonomous driving systems. Recent years have witnessed the emergence of a new paradigm that leverages *smart roadside infrastructure (SRI)* to enhance the limited compute and perception capabilities of standalone vehicles. In particular, several efforts [46, 47, 61, 74] have developed SRI-assisted perception solutions for autonomous driving. To date, the deployment of SRI is still in its infancy stage. Existing commercial products like roadside units (RSUs) [1, 42] and experimental prototypes [49, 56, 79] are standalone roadside systems that have limited compute and networking capabilities. Moreover, they still have a very low penetration rate and have not been fully validated in real-world deployments.

To realize the vision of infrastructure-assisted autonomous driving, SRI must address a multitude set of key requirements. First, as a shared infrastructure, SRI should serve a large number of vehicles and diverse autonomous driving applications. For example, it should provide passing vehicles on-road object detection results for informed driving decisions as well as point cloud segmentation results for the purpose of mapping and navigation [63]. Therefore, SRI needs to execute the

inferences of *multiple concurrent* deep learning (DL) models while meeting stringent real-time requirements. Second, SRI should be capable of both high-bandwidth infrastructure-to-infrastructure (I2I) and infrastructure-to-vehicle (I2V) communication. Data-intensive sensors like LiDAR and cameras have become the *de facto* configuration of commercial autonomous driving platforms [2, 8, 9]. These sensors produce high-precision data at a rate of tens of Mbps, which often needs to be shared between SRI nodes. However, to support such a high-bandwidth I2I communication at a large spatial scale, existing technologies like fiber optical Ethernet and 5G cellular networks would incur prohibitively high deployment and operational costs. Moreover, although several vehicle-to-everything (V2X) technologies are emerging, they are designed to achieve relatively low data rates [23]. Lastly, to lower the barrier of adoption, the existing traffic infrastructure such as lampposts should be reused or retrofitted as much as possible, which brings various physical and system challenges. For instance, the traffic lampposts typically have a limited power supply (i.e., $\sim$ 200 W in our city), which is challenging to power a fully functional SRI with advanced compute/communication capabilities.

A key contribution of this work is to explore a large design space of smart roadside infrastructure and identify a multitude set of key challenges that have not been addressed collectively by existing communication and computing technologies. We present *Soar*, the first end-to-end SRI system that is specifically designed for scenarios where the SRI can be deployed at a large spatial scale and operated inexpensively for a long period of time to support autonomous driving. Specifically, each *Soar* node comprises a low-power single-board edge computer, 802.11ac interfaces, and sensors chosen from three modalities (mmWave radar, LiDAR, and thermal camera), which carefully balances data quality, privacy protection, and power consumption. The communication architecture of *Soar* comprises a bi-directional multi-hop I2I network and a downlink I2V broadcast service. We exploit the naturally linear topology of the roadside lamppost network and adopt advanced network coding to achieve high-bandwidth and reliable multi-hop I2I communication. A novel I2V broadcast service is designed based on the injector-sniffer mode of 802.11ac, which incorporates lightweight channel switching and empirical measurement-based rate selection for realizing high-bandwidth broadcast to passing vehicles at high speed. To tackle the distinct computing challenges encountered in the development of SRI systems, *Soar* incorporates a hierarchical task management framework that facilitates desirable load balancing among nodes and enables efficient collaboration. Additionally, we have devised an opportunistic DL task scheduling mechanism to mitigate the impact of resource contention and dynamic system delays arising from multiple concurrent tasks on a single SRI node.

We deployed 18 *Soar* nodes on existing outdoor lampposts on campus, operational for over 2 years. We report experience in the design and deployment of *Soar*, as well as extensive experimental results. Our real-world testbed evaluation shows that *Soar* can support a diverse set of autonomous driving applications and achieve a 96.1% success rate of application-level data delivery, a 2× improvement over the state-of-the-art baselines. Compared to traditional 802.11ac-based approaches, the I2I and I2V communication system of *Soar* realizes high-bandwidth and reliable data transmission with 5× improvement in throughput over up to 9 hops and 3× improvement in broadcast bandwidth with multiple vehicles, respectively. Our findings and experiences in this work offer key insights into the development and deployment of next-generation smart roadside infrastructure and autonomous driving systems.

2 RELATED WORK

Smart roadside infrastructure. Several studies are focused on building real-world SRI with sensors and intelligent devices [49, 56, 79]. There are also several commercial projects [6, 19, 30, 31] on SRI development. However, these efforts are based on small-scale experimental deployments and do not address the comprehensive set of system challenges for autonomous driving. Other works [30, 33, 35, 38, 43, 44, 76] focus on roadside unit's software systems and vehicular networks but lack of real-world evaluation and deployment.

Infrastructure-assisted autonomous driving. Recently, leveraging sensors and compute units installed on roadside infrastructure to assist AVs has received significant attention. Several systems propose to process the sensor data on the infrastructure and provide application-level results to AVs such as object identification [67, 89, 92], landmark report [82], and parking assistance [32]. A number of studies [24, 26, 46, 47, 61, 73, 74, 88, 90] propose new methods of fusing perceptive information between vehicles or between the infrastructure and the vehicle. These studies focus on specific infrastructure-assisted technologies that shed light on the design objective of *Soar* in this work.

Concurrent DL task execution on cooperative edges. Several solutions have been proposed to optimize the performance of concurrent tasks on a single edge node, including on-device DL task scheduling and model compression [37, 54, 59, 60, 85]. However, they do not address the cooperation among edge nodes, which is essential for fully utilizing the limited resources of the entire edge system. Some studies focus on allocating DL tasks to the edge nodes in a workload-balancing manner [51, 86]. However, these approaches are designed for specific applications and are not applicable to AVs that require high real-time responsiveness.

Table 1: Cost and performance (Perf.) comparison between *Soar* and two alternative paradigms.

Module / Paradigm	I2I		Computing		I2V		Cost	
	Method	Perf.	Method	Perf.	Method	Perf.	Installation	Operation
Ethernet+Cloud	Fiber Ethernet	**High**	Cloud computing	**High**	5G V2X	**High**	High	Medium
5G+Cloud/Edge	5G cellular	Low/Medium	Cloud computing	**High**	5G V2X	**High**	Medium/High	High
Hybrid 5G/Ethernet+Cloud	Hybrid 5G/Ethernet	Medium	Cloud computing	**High**	5G V2X	**High**	High	Medium/High
Soar	802.11ac multi-hop	Medium	Collaborative edge	Medium	802.11ac broadcast	Medium	**Medium**	**Low**

3 OVERVIEW

3.1 Applications

Soar aims to enable a myriad of infrastructure-assisted autonomous driving applications that provide sensing information to vehicles in real time. Here we highlight three typical applications, each with representative characteristics that pose design requirements for *Soar*.

Percpetion sharing. A crucial application of SRI is to leverage sensors (e.g., LiDARs, etc.) to enhance the perception of autonomous vehicles (AVs). The information shared from the infrastructure to the vehicle can be raw sensor data [46, 47] or computed perception results [74]. AVs can integrate such information from the infrastructure with their own to construct comprehensive scenes and enhance their ability to understand and respond to complex driving environments. As a representative *data-intensive application*, perception sharing imposes the requirement for SRI to process and transmit a substantial volume of data to AVs in real time. For example, sharing LiDAR point clouds with AVs [47] requires a data transmission bandwidth over 30 Mbps (for a 32-line LiDAR [18]).

Traffic monitoring. Traffic flow monitoring [92] offers insights into traffic congestion and potential alternative routes that extend beyond the vehicle's visual range. As an essential requirement, it requires real-time data transmission between infrastructures when vehicles/pedestrians need to be tracked continuously over a period of time. Consequently, the infrastructure system must exhibit satisfying *scalability* and *cost-effectiveness* to support large-scale deployment and coordination. In addition, it underscores the need for *robust communication* between infrastructure nodes.

Accident warning. Accident warning [67] can aid AVs in detecting pedestrians and cyclists, especially under conditions of poor visibility or in areas with blind spots. As a life-critical application, it necessitates the infrastructure to deliver warning information to the vehicle timely. Considering that SRI may have to support multiple applications concurrently and mainstream perception-related applications are typically based on compute-intensive deep learning (DL) algorithms, it is essential for the SRI to handle concurrent DL tasks and meet their real-time requirements.

3.2 Design Objectives and Choices

Drawing upon the requirements derived from applications, we distill several design objectives that are essential in shaping the development and implementation of *Soar*, including 1) *high-bandwidth data communication* between infrastructure nodes as well as from infrastructure to vehicles; 2) *efficient inference of multiple concurrent DL models* with stringent realtime requirements; and 3) *low installation/operation costs*, e.g., by reusing/retrofitting the existing traffic infrastructure, for large-scale deployment.

We now discuss several design choices based on existing technologies and highlight the key challenges. Table 1 shows a comparative analysis of Soar's design choices against two other SRI paradigms.

Ethernet+Cloud. A straightforward paradigm of SRI is to connect all infrastructure nodes with specially laid fiber optical Ethernet and offload all computing tasks to the cloud. SRI can communicate with vehicles via 5G V2X, the most advanced V2X technology currently available [65]. This paradigm can deliver robust task execution performance facilitated by high-speed and reliable Ethernet connections as well as the cloud infrastructure. However, it necessitates substantial investment on fiber optical cable installation and cloud servers, particularly in rural regions that lack such network and computing infrastructures. In addition, existing V2X technologies (e.g., LTE-V2X) focus on low data rates (up to 25 Mbps [23]) and hence cannot handle large data volumes like LiDAR point clouds in real time. Moreover, V2X devices on the market remain expensive. For instance, an off-the-shelf Gohigh 5G-V2X transmitter-receiver pair costs over 15k USD [42].

5G+Cloud/Edge. Another alternative paradigm is to offload computation to the cloud while the nodes communicate with the cloud via a 5G cellular network. Compared with the *Ethernet+Cloud* paradigm, this 5G cellular approach demands merely a low-cost 5G interface for each infrastructure node. However, the operational expenses of such a system are notably higher due to the service charges of 5G data plans. For example, a LiDAR-equipped SRI node may need to transmit more than a TB of data via 5G during a single day. Furthermore, in areas such as cities with densely deployed roadside SRI nodes, transmitting large volumes of data from SRI to the cloud through a 5G cellular network can incur significant network overhead, thereby adversely impacting application

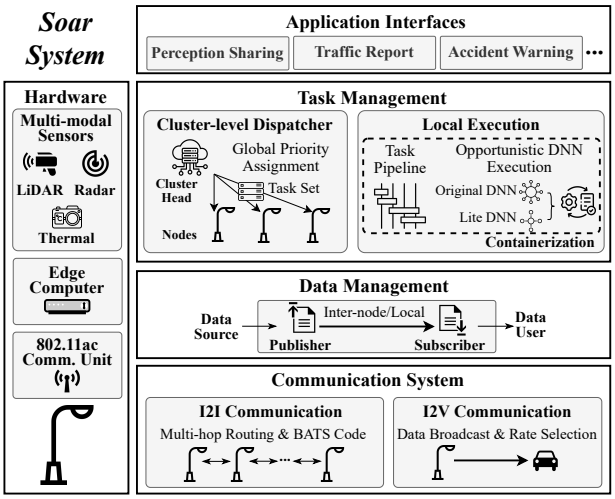


Figure 1: The system architecture of *Soar*.

performance. Another approach within this paradigm is to leverage the edge computing capability of 5G cellular networks [78]. It typically involves the deployment of dedicated 5G base stations and edge servers specifically designed to execute computing tasks from SRI. Nevertheless, this approach necessitates the installation of numerous additional base stations as well as high-speed network connectivity between them to handle the large amount of data from SRI nodes, which incurs considerably high deployment costs.

Hybrid 5G/Ethernet+Cloud. A hybrid paradigm of SRI leverages Ethernet for high-speed connections between infrastructure nodes. With computing tasks still offloaded to the cloud, it integrates 5G for the communication between infrastructure nodes and the cloud. This paradigm features reliable data transfer and 5G's expansive reach. However, it still incurs significant initial costs for fiber optical Ethernet setup and recurring expenses for 5G data plans.

3.3 System Architecture

Soar adopts a systematic design methodology that seamlessly integrates I2I and I2V wireless communication, efficient data/compute task management, and careful power-efficient hardware design. *Soar* achieves a remarkably low total power consumption of approximately 70W, making it easily deployable on existing traffic lampposts without requiring extensive upgrades or modifications[1]. We note that the design choices made by *Soar* target specifically the scenarios where SRI needs to be deployed at scale and operated with minimal expense. However, given the highly diverse real-world settings, different technologies would likely coexist and complement each other during the adoption and deployment of SRI.

[1]Most operational traffic lampposts have a power supply budget of about 100 ∼ 200 W, most of which is used for lighting [3].

Fig. 1 shows the system architecture of *Soar*. It comprises a series of roadside infrastructure nodes, each equipped with various sensors, an edge computer, and communication interfaces. The *Soar* nodes can communicate with each other via a multi-hop wireless network and broadcast to passing vehicles. *Soar* adopts a hierarchical task management framework to first allocate DL tasks within a cluster of nodes and further optimizes their concurrent execution on each node at runtime. Moreover, we design a set of application interfaces to support diverse autonomous driving applications.

Hardware system. We choose three sensor modalities for *Soar*: millimeter-wave (mmWave) radars, LiDARs, and thermal cameras. MmWave radars are used for background tasks such as speed measurement and vehicle counting. LiDARs capture high-resolution yet privacy-preserving 3D information about scenes and objects on the road. Motivated by the fact that *Soar* only requires a view of the road, we employ two low-cost solid-state LiDARs with an FOV of around 90° facing opposite directions of the road. Compared to traditional omnidirectional LiDARs [18], our design leads to a 10× cost reduction. Thermal cameras are chosen for their night vision capability and the preservation of user privacy, although RGB cameras can also be easily integrated. Each *Soar* node equips a single-board edge computer as the main control and compute unit. The communication subsystem of *Soar* requires fairly high processing capability to implement reliable high-bandwidth I2V and I2I communication (see § 4). Therefore, it is implemented using a standalone embedded computer board. Both the sensors and communication unit are connected to the edge computer via a PoE switch. Such a modular design improves both the system robustness and the cost-effectiveness of part replacement. Moreover, each *Soar* node achieves a total power consumption of ∼70 W.

Communication system. The communication architecture of *Soar* comprises a bi-directional multi-hop I2I network and a downlink I2V broadcast service. Specifically, we choose to implement I2I communication using a wireless 802.11ac network in a multi-hop manner and adopt advanced network coding to achieve high bandwidth and reliability. 802.11ac can leverage existing roadside infrastructure and consumer-level equipment, lowering upfront and operational costs. The multi-hop approach allows to extend the network's coverage without needing extra infrastructure. We then adopt an I2V communication framework based on the injector-sniffer mode of 802.11ac that focuses on unidirectional downlink broadcasting from *Soar* to vehicles. *Soar* focuses on passive broadcasting to significantly ease the integration of SRI with existing autonomous driving systems. Applications such as perception sharing and accident warning only need vehicles to receive and process sensor information from the infrastructure, making the passive broadcast model an ideal choice

as vehicles in the vicinity of each other usually require identical information about the surroundings.

Data and task management. For data management, we adopt a publisher-subscriber scheme where *Soar* creates a data publisher for each sensor and launches subscribers to fetch data. This design decouples the data consumption from sensors, thus achieving the plug-and-play of multi-modal sensors at the software level. Moreover, it prevents sensors from direct access by downstream tasks, which enhances the security of sensor data.

Soar adopts a collaborative edge approach [52] and proposes a novel hierarchical task management architecture that distributes data processing and tasks across local Soar nodes. Thus, Soar can leverage the processing capabilities of local infrastructure nodes so that tasks can be processed closer to the data source to reduce data transmission and latency. *Soar* nodes are first clustered based on their geographical distribution (e.g., the nodes on the same road section are grouped into the same cluster). The cluster head dispatches tasks within the cluster by jointly considering task urgency and resource availability of nodes. Such a cluster-level task allocation approach achieves desirable load balancing among nodes and enables them to collaborate efficiently to run multiple data-intensive applications. Moreover, it naturally supports location-awareness of data and compute tasks, such as processing sensor data from nodes around busy intersections and sharing results with vehicles ahead of time. In addition to the cluster-level task dispatching, we design an opportunistic DNN execution mechanism for local task execution on each *Soar* node. Our key idea is to generate a lite DNN model for each DL task, and then choose one of the versions to execute at runtime, which can accommodate highly dynamic communication bandwidth and computing resources.

Application interfaces. We design a set of application interfaces to efficiently share the sensor data or compute results with vehicles. First, *Soar* supports the provision of raw or processed sensor data based on the requirements of different downstream autonomous driving applications. Using LiDAR data as an example, *Soar* broadcasts three levels of LiDAR results: the raw point clouds, semantic segmentation results, and object detection results. They can be utilized by mapping [61], navigation [63], and decision-making [72] tasks on the vehicle, respectively. Moreover, *Soar* can also broadcast high-level traffic information such as current traffic congestion level or accident warnings to surrounding vehicles.

4 COMMUNICATION SYSTEM

4.1 Multi-hop I2I Network

A key design objective of *Soar* is to achieve high-bandwidth multi-hop I2I communication, i.e., about 100 Mbps over up to 10 hops within 500 meters. This allows the *Soar* nodes located on the same road section to achieve efficient coordination and load balancing. As depicted in § 3, *Soar* leverages a cluster-level task allocation approach to overcome the limited compute resources on each node. Such task offloading requires high-bandwidth and reliable data transmission among *Soar* nodes. For instance, a typical LiDAR produces point clouds at a rate of more than 30 Mbps [18]. As discussed in § 3.2, we opt to employ off-the-shelf 802.11ac to implement I2I communication of *Soar* for its wide availability and cost-effectiveness. The left part of Fig. 2 illustrates our I2I communication architecture.

Linear network topology and multi-hop routing. The mesh topology is widely adopted in multi-hop wireless ad hoc networks [53, 69]. However, it is well known that the throughput of wireless mesh networks is low in real-word settings due to severe interference and channel contention. The design of *Soar* addresses this challenge by exploiting the naturally linear topologies of roadside nodes. Specially, we implement a multi-hop routing strategy where each node has two bridged 802.11ac interfaces operating in AP and STA modes, connecting to previous and next nodes. This design offers greater scalability compared to Ethernet or 5G cellular networks in terms of cost-effectiveness and deployment flexibility. In case of link failures, nodes automatically attempt to establish a connection with a further node. We adopt such a semi-fixed routing strategy instead of fully dynamic multi-hop routing protocols [29, 34, 70] because the link between adjacent *Soar* nodes typically have a line of sight and short distance (e.g., 30 $\sim$ 50 m [28, 68]), resulting in good link quality.

Network coding. Multi-hop wireless networks often suffer from significant packet loss [22]. While various link layer techniques, such as network coding [21, 58] and fountain codes [62, 75], have been proposed to enhance multi-hop reliability, the high compute and storage costs limit their applicability to resource-constrained edge platforms [84, 87]. To tackle this challenge, we employ the Batched Sparse (BATS) code [83], an advanced network coding approach that addresses the packet loss in linear multi-hop networks with extremely low power and storage consumption. We exploit it to optimize end-to-end communication throughput and relieve heavy retransmission overhead. Specifically, BATS code features an outer code and an inner code, with the outer code being a matrix generalized fountain code generating numerous batches for high capacity. During the multi-hop transmission, an inner code derived from random linear network coding [50] is applied to the batched packets at each relay hop, realizing persistent reliability with low overhead. Though several studies [84, 91] have demonstrated the performance of BATS code in simulations or lab settings, *Soar* is the first system to apply BATS codes in real-world I2I communication. We implement BATS codes at the network layer

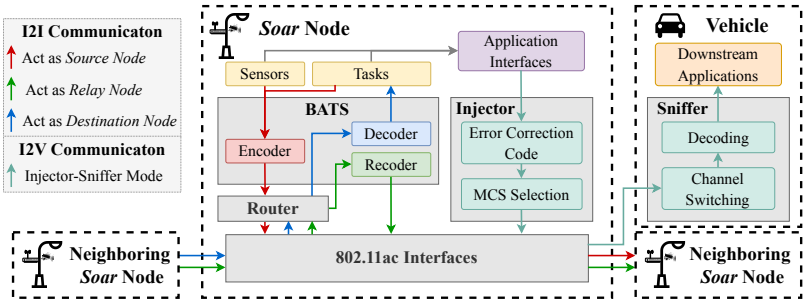


Figure 2: Communication system architecture of *Soar*.

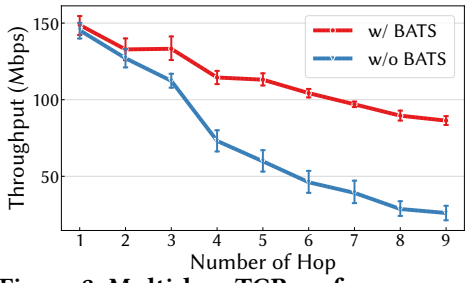


Figure 3: Multi-hop TCP performance w/ and w/o BATS code.

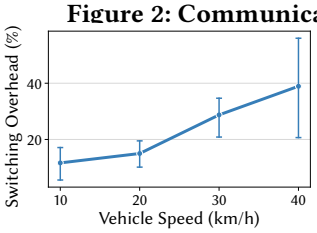
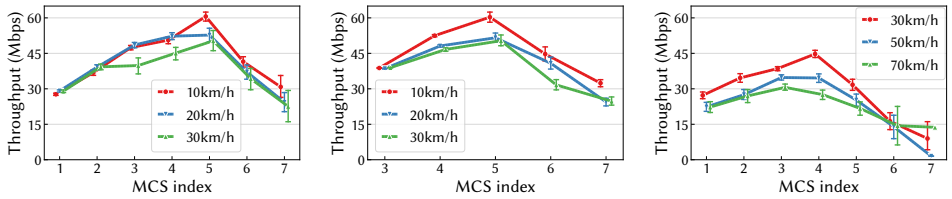


Figure 4: 802.11ac switching overhead.



(a) Campus road. **(b) Community road.** **(c) Public road.**

Figure 5: Measurements of MCS performance in different road environments where 802.11ac with 2 spatial-stream is applied.

to be compatible with the existing network stack, which is transparent to traditional socket and transport layer protocols. We utilize *Netfilter* [16] to capture IP packets and apply different coding depending on their destinations. Fig. 3 shows *iperf3* [14] results of a 10-node wireless linear network. With BATS codes, TCP throughput can achieve above 100 Mbps over 6 hops and 90 Mbps over 9 hops, an up to 5× improvement compared to the baseline without BATS codes.

4.2 I2V Broadcast

Soar needs to transmit large volumes of data to passing vehicles for downstream autonomous driving tasks, such as providing LiDAR point clouds for 3D perception, which demands ∼ 30 Mbps bandwidth [47]. *Soar* aims to achieve a communication bandwidth of ∼ 50 Mbps between SRI and multiple vehicles.

Injector-sniffer-based high-bandwidth broadcast. As discussed in § 3.2, we propose an 802.11ac-based I2V communication framework for high-bandwidth, cost-effective data broadcast from each *Soar* node to nearby vehicles. Our design focuses on the unidirectional downlink broadcasting and utilizes the injector-sniffer mode [80, 81] to achieve high data rates without link establishment. *Soar* provides either raw sensor data or detection results (e.g., bounding boxes), which can support a range of autonomous driving tasks that rely on SRI data to enhance perception and reliability. The *Soar* node acts as an injector, transmitting raw wireless packets with headers into the air, while vehicles sniff packets using onboard 802.11ac receivers. This broadcast design without vehicle feedback makes *Soar* highly scalable with respect to the number of vehicles in the network. The right part in Fig. 2 shows the I2V communication pipeline. *Soar* node first encodes data with error correction, then modulates it using an empirical rate selection scheme, and finally transmits on a pre-assigned channel. Vehicles decode packets, recover original data for applications, and determine the channel to receive messages based on location or signal strength.

Error correction code. Without vehicle feedback, our passive data broadcast design may experience severe packet loss, hindering goodput for downstream applications. To tackle this challenge, we apply a systematic random linear code [40] for error correction, which generates multiple check packets on application layer to ensure high data delivery reliability.

Measurement-based rate selection. After applying error correction, *Soar* modulates data for transmission. A challenge in designing the modulation and coding scheme (MCS) is selecting the optimal rate for reliable high-throughput transmission. Existing works on the multicast rate selection rely on receiver feedback and thus are incompatible with our passive design. Our extensive real-world experiments show that there exists a trade-off between modulation rate and packet loss, depending on the environmental dynamics. Fig. 5 illustrates that there exists an optimal MCS that demonstrates consistent performance under different vehicle speeds but varies with the road environments. This is because the impact of the environment (i.e., road shape, trees, buildings, etc.) dominates the performance of I2V communication under different MCSs, while the impact of low urban speeds (i.e., < 70 km/h) is not significant. Motivated by this observation, *Soar* adopts an empirical MCS rate selection scheme where each node is configured with a fixed rate based on installation-phase measurements.

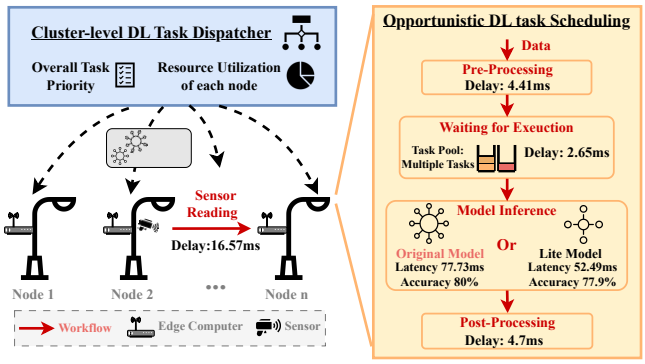


Figure 6: The task management framework of *Soar*.

Lightweight channel switching. Unlike 802.11ac incurring significant reassociation overhead at high vehicle speeds (Fig. 4), our design only requires vehicles to switch sniffing channels to receive data from *Soar* continuously. With 24 non-overlapping channels in the 5GHz band, each *Soar* node can be assigned a different channel without inducing the interference among the same cluster. When a strong interference level (e.g., caused by nearby Wi-Fi, 5G signals, etc.) is detected on the assigned channel, *Soar* node will switch to another free channel. Each *Soar* node will periodically broadcast the channel assignments for its cluster on a prefixed control channel (e.g., channel 36). Passing vehicles first listen to the control channel for channel assignments, and then switch channels during movement based on their own and nearby *Soar* nodes' locations [2].

Security Issues. Without association and uplink from vehicles to the infrastructure, our passive I2V broadcast design may be vulnerable to security issues like malicious data injection. To mitigate such attacks, we employ private key encryption [36] to create a digital signature within data packets. Certified vehicles can obtain a public key, which is used to verify the signature upon data receipt to ensure security.

5 TASK MANAGEMENT

A key challenge of *Soar*'s design is supporting executing multiple real-time DL tasks concurrently on resource-constrained edge platforms. We propose a novel task management framework that can efficiently dispatch DL tasks among multiple *Soar* nodes in a collaborative manner. Our design is motivated by the following characteristics of roadside infrastructure nodes. First, the DL tasks on different nodes are highly diverse, due to different road sections and heterogeneous hardware/sensor configurations. For instance, the *Soar* nodes at the crossroads are typically installed with more types of sensors for complex tasks such as vehicle tracking and pedestrian detection. Second, the communication bandwidth and compute resource available on each *Soar* node are highly dynamic. For instance, traffic spikes during rush hour may lead to a fluctuation in communication bandwidth.

To address these challenges, as shown in Fig. 6, our task management framework is based on a two-tier clustering structure, where the *Soar* nodes can be naturally clustered based on road sections or geographic locations. First, *Soar* employs a cluster-level task dispatcher, which is responsible for balancing task workloads in a cluster by jointly considering task priorities/deadlines, resource availability, and geographical positions of nodes. Second, *Soar* includes a local opportunistic task scheduler to control the execution of multiple DL tasks on a single *Soar* node, which aims to optimize the real-time performance of concurrent DL tasks under runtime system dynamics.

5.1 Cluster-level DL Task Dispatching

We design a cluster-level task dispatcher for SRI based on both overall task priorities and resource utilization of each *Soar* node. Our goal is to maximize the number of total deployed tasks while meeting their real-time requirements, as shown in Eq. 1. $Priority(m)$ denotes the task priority weight of task τ_m. $N(\tau_m, n)$ equals 1 if the task τ_m is deployed on the Node n, else it equals 0. $T^{E2E}(\tau_m, n)$ denotes the end-to-end delay of the task τ_m executed on Node n, which is defined as the total delay between the launch and the completion of a task. T^{E2E} also includes the communication time, which is estimated based on the periodically measured bandwidth. $DDL(\cdot)$ is the expected completion time (i.e., deadline) of each task, which is not less than the trigger period $Period(\tau_m)$ of the task τ_m.

$$\begin{aligned} \max_{\forall m} \quad & \sum_{m=1}^{M} \sum_{n=1}^{N} Priority(m) \times N(\tau_m, n) \\ \text{s.t.} \quad & T^{E2E}(\tau_m, n) \le DDL(\tau_m), \sum_{n=1}^{N} N(\tau_m, n) \le 1 \end{aligned} \tag{1}$$

An exhaustive search of the problem formulated in Eq. 1 has a complexity of $O(MN)$. We adopt an efficient heuristic as follows. First, we sort the task pool from high to low priority according to the ascending order of $1/deadline$. We also map the task priorities to a sequence of power of 2. After this mapping, the task with the longer relative deadline is prioritized as the lower priority 2^0. In this way, the priority weight $Priority(m)$ of tasks with high priorities can dominate those with low priorities. Then, we determine the deployed node for each task from high to low priority weight.

For each task, we search for its deployment node from its source node to other adjacent nodes based on the physical proximity. If the time constraint in Eq. 1 is met, we search for a *Soar* node that has the lowest resource utilization to deploy the task τ. The resource utilization of a Node n can be estimated by the sum of the task laxity of all the tasks on this node, where the laxity for task τ_m is defined as

[2]The locations of *Soar* nodes are easily accessible by vehicles since they are installed on existing lamppost infrastructure. *Soar* nodes may also obtain their own locations through GPS and broadcast to vehicles.

$T^{total}(\tau_m, n)/Period(\tau_m)$. $T^{total}(\tau_m, n)$ denotes the total execution time of the task τ_m on Node n, which contains sensor reading, task pre/post-processing and model inference.

5.2 Opportunistic DL Task Scheduling

Due to severe resource constraints and significant dynamics in the real world, it is challenging to support multiple real-time DL tasks concurrently on the edge platform of the *Soar* node. As shown in Fig. 6, an end-to-end DL task contains sensor reading, pre-/post-processing, blocking, and model inference. Significant dynamics in the wild such as power surges will cause unpredictable delays in sensor reading. Moreover, resource contention between different system processes (e.g., sensor reading, CPU/GPU processing etc.) will cause dynamic blocking delays. To address these challenges, *Soar* adopts an opportunistic execution mechanism to mitigate the impact of fluctuating system delays on the inference of multiple concurrent tasks.

Our main idea is to generate a *lite model* for each DL task, which can better adapt to unpredictable resource availability at runtime. The lite model can be a compressed DNN model or a small model that distillates knowledge from the original model. However, the lite model achieves a desirable low latency at the expense of accuracy. To mitigate the potential accuracy loss, we design an online opportunistic scheduling algorithm, which executes the lite models only when the time constraints cannot be met, thus minimizing the impact on task accuracy. By exploiting such model compressibility, *Soar* can reduce the resource demand of each inference according to the runtime condition. *Soar* preloads both the original and the lite models during initialization to avoid dynamic loading delays. Our design can also work with existing storage optimization methods such as weight sharing [37, 60] to reduce the storage overhead of lite models.

Given a task set allocated by the cluster head, we split each task into pre-/post-processing and model inference. These task segments are then assigned to different processes and executed in a pipelining manner. This design enables full utilization of both CPU and GPU resources by executing pre/post-processing for one task alongside the inference of other tasks in parallel. At runtime, we first determine which model to be executed according to the urgent level of each model inference. Specifically, *Soar* chooses the model inference job with the latest deadline in the queue for execution. *Soar* then determines the DNN model (i.e., the original or lite model) to be executed according to the remaining time of the current and the next inferences. We estimate the completion time of each model inference through the measurement results from offline profiling. Meanwhile, *Soar* checks one more job in the queue. If there is not enough remaining burst time for the next urgent inference, *Soar* also uses the lite model for the current inference. The jobs that miss the deadline are dropped before execution to prevent the delay accumulation.

6 IMPLEMENTATION AND DEPLOYMENT

6.1 System Implementation

We deployed 18 *Soar* nodes on existing lampposts in two clusters on our campus, as shown in Fig. 7. The first cluster consists of 6 nodes around a parking lot, with a server inside a building serving as the cluster head. The second cluster has 12 nodes along the main campus road and two cluster heads in buildings. The two clusters cover 0.3 km and 0.5 km of road, respectively. Our campus testbed has received approval from the Institutional Review Board (IRB).

Fig. 8a shows the prototype of *Soar* implementation. Each *Soar* node is mounted at the bottom of a lamppost, whose components and layout are shown in Fig. 8b. The system uses an NVIDIA Jetson TX2 computing board [66] in a waterproof enclosure, and other devices such as the communication box and sensors are connected to it via a PoE switch. We implement the communication system in a separate industrial single-board computer whose hardware is modified to support multiple 802.11ac network cards. To support I2V broadcast, we modified the RTL8814AU [7] driver to enable customized packet injection. Most nodes are equipped with a Teledyne FLIR thermal camera [39] and two Livox Horizon LiDARs [13], and some also have a TI mmWave radar [15]. Two or three directional antennas are mounted on the top, pointing at adjacent nodes, and a set of I2V antennas is installed below, facing the road. Each node costs 10.2k USD approximately, with the sensors costing around 2.75k USD and other components costing 1.95k USD. We implement the sensor data management module based on ROS2 [64], which offers various sensor drivers and APIs for realizing the publisher-subscriber mode data management. The task management module is implemented in KubeEdge [71], which allows the cluster head to efficiently dispatch task codes and their environmental dependencies to each node through the containerization technique. To periodically estimate the I2I communication bandwidth between *Soar* nodes during cluster-level task dispatching, we use *iperf3* [14] to assess bandwidth on each link every 10 minutes, which incurs negligible overhead.

6.2 Deployment Experience

Installation considerations. To save lighting power during day time, the lampposts on campus are powered on/off automatically by a solar timer-controlled switch on the circuit bus. To power *Soar* nodes from lampposts continuously, we removed the switch on the bus and modified the power supply on each lamppost into two parts, one nonstop power supply

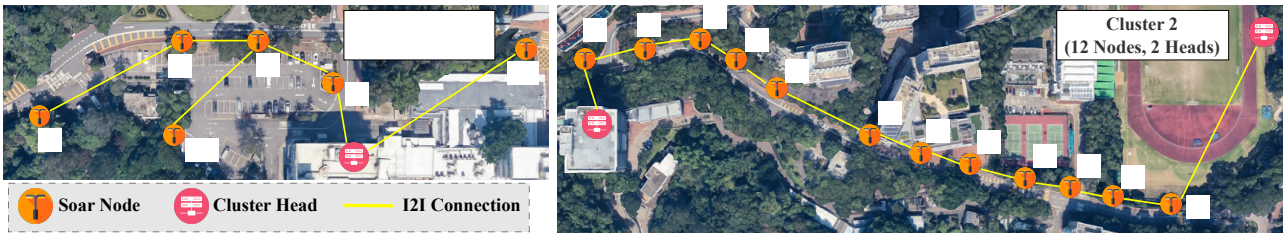
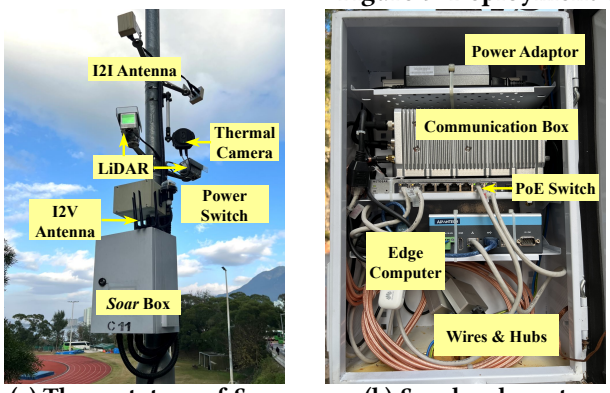


Figure 7: Deployment of two *Soar* clusters on campus.



(a) The prototype of *Soar*. **(b) *Soar* box layout.**

Figure 8: Hardware system of the *Soar*.

for each *Soar* node and a solar timer-controlled one for lighting. Moreover, by calculating the load-bearing constraints in extreme weather conditions, we limit the installation height below 5 m, the weight below 30 kg, and the lantern windage area below 0.12 m^2. As a result, we installed the antennas and sensors above 3 meters on the lamppost due to their light weight (i.e., around 5 kg), which avoids any occlusion to the view of the road.

System durability and robustness. We also optimized the durability and robustness of *Soar* for in-the-wild deployment according to more than two years of operational experience. We added a shelf to hold the box of computing devices and store the waterproof cables at the bottom, which can effectively avoid water accumulation over time. Moreover, during the long-term operation of *Soar*, we observed that the lamppost power supply could occasionally fall short of the operational power requirement of *Soar*, causing a system reset. This is due to either dynamic grid conditions or the high instantaneous power draw at the startup of hardware devices. For example, Jeston AGX Xavier [12] consumes 65 W of power at startup but only up to 30 W at runtime. This problem can be addressed by adding an uninterruptible power supply, which has been validated on our system.

Edge software architecture. To realize the containerized software deployment, we initially chose Kubernetes (K8S) [17] due to its support for container management. However, during our deployment, we found various issues, such as the unstable connection between *Soar* nodes and the cloud server (i.e., via TCP socket). In this case, the container deployment from the server to the *Soar* nodes can be terminated or even forcibly removed by K8S. Therefore, we switched to KubeEdge [71] as our development framework, enabling the autonomous edge operation even during disconnection from the cloud. Furthermore, the KubeEdge-based implementation occupies only up to 80 MB of memory, a 60% reduction compared to the K8S-based implementation.

7 EVALUATION

7.1 Evaluation Methodology

Field studies and self-collected dataset. We conduct extensive experiments based on the two clusters of *Soar* nodes on campus (see § 6). We drive a test vehicle equipped with a LiDAR, a GPS, a four-antenna array, and a Jetson Orin (see Fig. 9) to continuously receive data from *Soar* nodes. To simulate the presence of other vehicles receiving data from *Soar*, we randomly placed some fixed 802.11ac devices as pseudo vehicles receiving data from nearby *Soar* nodes to simulate different traffic conditions. Moreover, we collect infrastructure- and vehicle-side LiDAR point cloud datasets (4250 frames in total) and manually annotate them to train models for evaluating application performance on *Soar*.

Application implementation. We implement three typical autonomous driving applications that require executing DL tasks on *Soar*, which are shown in Table 2. These applications include LiDAR-based perception sharing, RGB camera-based traffic monitoring, and thermal-camera-based jaywalk warning. Due to the privacy concerns of the campus testbed, we deploy a virtual camera in our data management framework that streams KITTI data [41]. For both traffic monitoring and jaywalk warning, we choose YOLOv5s-based and YOLOv5n-based models as the original and lite models. These models are trained using the KITTI dataset and the Teledyne FLIR ADAS dataset [39], respectively. In the perception sharing, we compress the original PointPillars [57] with width/depth scaling methodology [77] to generate a lite model, and train them with our self-collected dataset. These three applications run concurrently on each node for all the experiments, and the results are transmitted to the vehicle.

To further evaluate the benefits of *Soar* for AVs, we implement two typical infrastructure-assisted perception fusion applications: point cloud registration and LiDAR perception extension. In point cloud registration, the vehicle receives

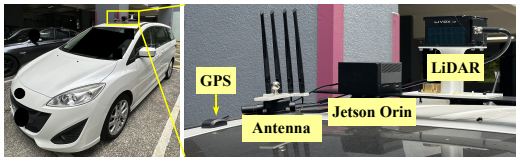


Figure 9: The test vehicle.

raw LiDAR point clouds and aligns them with its own point clouds. The LiDAR perception fusion is the downstream application of perception sharing, where the vehicle receives the object detection results and fuses them into its view.

Metrics. For evaluation of the communication system of *Soar*, we are interested in *throughput*, *switching overhead*, and *packet delivery ratio (PDR)*. The switching overhead is measured by calculating the fraction of time spent on switching the channel during receiving data from a *Soar* node. To evaluate the task management performance, we quantify the real-time performance achieved by our framework using the *deadline missing rate*, a widely-used metric for real-time performance evaluation [27, 60]. Specifically, the deadline missing rate contains the job drop and exceed ratios, which indicate the ratios of jobs dropped because of overdue and failing to meet their deadlines, respectively. We also statistic the *average end-to-end (E2E) delay* (defined in § 5) of each job to quantify the execution efficiency of the DL tasks. Moreover, to focus on the overall performance of *Soar* in the applications, we define a *failure rate* to indicate the percentage of cases where the system failed to deliver application data to the vehicle. The failure cases include failing to execute in time and failing to transmit the results to the vehicle.

Baselines. We compare both our I2I and I2V communication with the traditional 802.11ac approaches. Specifically, we evaluate the I2I communication performance w/ and w/o BATS code. Besides, the I2V communication of *Soar* is compared with traditional 802.11ac broadcast and unicast modes. For comparison of our task management framework, we implement a task execution mechanism called Local-EDF which executes the DL tasks locally on their source nodes and adopts the Earliest Deadline First (EDF) scheduling policy for concurrent DL task execution. For the evaluation of the overall performance of *Soar*, we combine the 802.11ac unicast approach and the Local-EDF task management scheme as the baseline. We also compare *Soar* with three cloud-based baselines. In addition to the settings outlined in Table 1 (referred to as Ethernet+Cloud and 5G+Cloud), we introduce an additional baseline called 802.11ac+Cloud. This baseline replaces the Ethernet/5G I2I method with campus Wi-Fi.

7.2 Overall Performance

Evaluation setup. We evaluate *Soar* with varying combinations of applications across nodes of Cluster 2. *Soar* nodes have varying task sets since they have different combinations of sensors. Table 2 presents the distribution of tasks on nodes. The *source node* is where the data required by the task

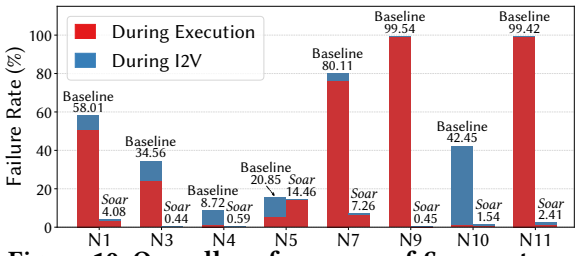


Figure 10: Overall performance of *Soar* system.

is generated, which is set according to the sensors and the covered road condition (e.g., crossings). *Soar* aims to extend the field of view for AVs, enabling them to have a broader perception. The *Soar* node shares application results with the vehicle when it enters the sensing range. We set the *deadline* for the delivery of results so that the vehicles can receive a considerable portion of the SRI's view. As an example, we now derive the deadline setting for a vehicle entering the sensing range of a *Soar* node (e.g., 50 m sensing range) at a speed of 70 km/h. We assume that the *Soar* node must share at least 90% of the sensing range (i.e., 45 out of the 50-meter range). That is, the data from *Soar* must be received by the vehicle when it travels no more than 5 meters into the sensing range of *Soar* node, which is equivalent to a time duration of $\frac{5\,m}{70\,km/h} = 250\,ms$. In our experiments, the sensors on *Soar* typically have a sensing range of $50 \sim 70\,m$. To ensure that vehicles traveling at speeds between $50 \sim 70\,km/h$ share at least 90% of *Soar* node's sensing range, we set application deadlines in the range of $250 \sim 500\,ms$. Specifically, we set three different levels of application deadlines as outlined in Table 2, with the most urgent deadline set at 250 ms for perception sharing and jaywalk warning, and 300 ms for traffic monitoring, as the former applications typically require more urgent attention. We note that *Soar* essentially functions as an on-the-air sensor for vehicles to provide additional perception beyond the short range around the vehicles. Since most autonomous vehicles only rely on their own sensors to perceive the short-range surroundings, *Soar* does not affect the safe-critical short-range path planning.

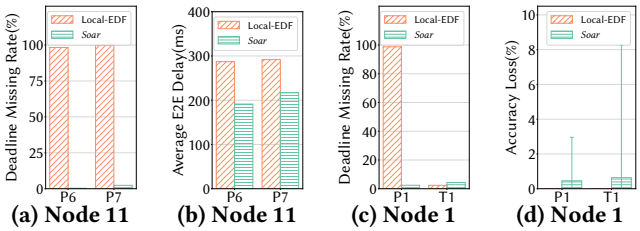
Overall task performance. We evaluate the overall application performance by comparing *Soar* with the overall baseline described in § 7.1. We provide a comprehensive analysis under the most urgent deadline setting, and summarize the performance improvement achieved for the other two deadline settings. Table 3 shows the allocation strategy generated by *Soar*. We observe that *Soar* successfully offloads the DL tasks from the heavily loaded node to the idle or lightly loaded node. For example, a data-intensive PointPillars-based DL task migrated from Node 11 to Node 12. Fig. 10 shows the failure rate for each *Soar* node. We omit several *Soar* nodes in Fig. 10 as we calculate the failure rate based on tasks' source node, and no task is sourced from these omitted nodes. We observe that *Soar* can achieve a failure rate

Table 2: Application settings and distribution. Text in italics describes the settings for lite models.

Application	Source Node	Model	Sensor	Execution	Deadline (ms)	Accuracy
Perception Sharing (P)	1,3,7,9,11	PointPillars	LiDAR	128.0 /*118.3* ms	250 / 300 / 350	85.5% / *83.0%*
Traffic Monitoring (T)	1,2,5,7,10	YOLO	Virtual Camera	37.2 /*32.7* ms	300 / 350 / 400	87.1% / *81.0%*
Jaywalk Warning (J)	3,4,5	YOLO	Thermal	78.3 /*43.3* ms	250 / 300 / 350	84.7% / *82.0%*

Table 3: Task allocation results on smart lampposts.

	Node1	Node2	Node3	Node4	Node5	Node6	Node7	Node8	Node9	Node10	Node11	Node12
Baseline	P1,T1	/	P2,J1,J2	J3,J4	T2,T3,J5	/	P3,P4,T4	/	P5	T5,T6	P6,P7	/
Soar	P1,T1	J1	P2,J2	J4	J3,J5	T2,T3,T4	P3	P4	P5	T5,T6	P6	P7



(a) Node 11 (b) Node 11 (c) Node 1 (d) Node 1

Figure 11: Performance of task management.

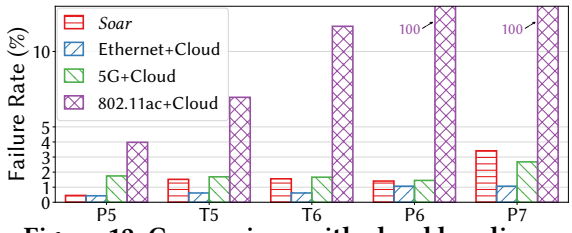


Figure 12: Comparison with cloud baselines.

reduction at 41.19% on average before transmitting the application results. After transmitting the application results, *Soar* can maintain a failure rate below 14.46% (3.90% on average), while the baseline incurs a failure rate up of 99.54%. In summary, *Soar* reduces the failure rate significantly (50.93% on average) compared with the baseline. For the other two deadline settings, *Soar* demonstrates an average reduction in the failure rate of 28.39% and 24.40%, respectively. The results show that *Soar* consistently maintains a reliable application performance for AVs among multiple *Soar* nodes.

We further analyze the advantages of task management in *Soar* by focusing on the real-time performance of each task on a single node. Fig. 11 shows the deadline missing rate and average E2E delay of each task sourced from Node 11. *Soar* reduces deadline missing rate efficiently by 97.67%. The average E2E delay is decreased by 85.43 *ms* since our task management has successfully allocated the task *P7* from heavily loaded Node 11 to the idle Node 12. We also show the deadline missing rate and accuracy loss on Node 1. The results show that *Soar* can reduce the deadline missing rate by 47.16% on average with all tasks executed locally. This is because our opportunistic DL task scheduler efficiently chooses the lite model for execution when the remaining time for the current and next inference is about to run out, which only leads to 0.55% accuracy loss on average.

Performance comparison with cloud baselines. We evaluate the performance between *Soar* and cloud baselines (see § 7.1) using different network configurations. Since our outdoor testbed supports limited communication settings, we set up an indoor testbed that supports the simulation of diverse network conditions. Specifically, three TX2 edge nodes transmit their sensor data to an NVIDIA GeForce RTX 2080 Ti server for processing through a wired Ethernet connection. We implement the baselines for cloud-based implementations by applying a trace collected from a public local AWS [11] server using *CloudPing* [10], in which the result exhibits a 15 *ms* round-trip time (RTT) between *Soar* nodes and the cloud on average. To emulate the 5G+Cloud and 802.11ac+Cloud communication, we capture real-world traces using *SpeedTest* [4] at each *Soar* node of our outdoor testbed. These traces contain bandwidths and dynamics of 5G and campus Wi-Fi. We replicate the settings used for Node 9, 10, 11 from Fig. 10 and show the failure rate of each task in Fig. 12. The results indicate that Ethernet+Cloud outperforms *Soar* in terms of performance but comes with higher deployment costs, as discussed in § 3.2. However, *Soar* demonstrates an average reduction in failure rate of 0.17% and 42.85% compared to 5G+Cloud and 802.11ac+Cloud, respectively. 802.11ac+Cloud performs poorly because many tasks failed to meet the deadlines due to significant delays caused by the transmission of large volumes of raw sensor data. Although 5G is capable of achieving high-bandwidth communication, our traces indicate that its uplink performance was unsatisfactory, only reaching 80 *Mbps* on average. Furthermore, we observe that the network may suffer significant bandwidth degradation (i.e., 40 *Mbps*) due to various blockages. Such limited bandwidth leads to significant delays, resulting in a high failure rate. Besides performance gain, *Soar* incurs lower operation costs (c.f., Table 1). Although cloud-based alternatives reduce the maintenance cost of computation units, the costs associated with repairing or replacing faulty sensors and communication devices are inevitable. In contrast, *Soar* avoids costly 5G operations, base station installation, and cable maintenance, making it a viable and practical solution for long-term deployment.

Performance of applications on the vehicle. We evaluate the *Soar* can support infrastructure-assisted autonomous

Table 4: Point cloud registration on *Soar*.

Methods	Offline	*Soar*	*Soar* w/o ECC
Registration Error (m)	0.10	0.16	0.39
Success Rate (%)	85.32	77.31	52.43
Density Benefit (pts/m^2)	46	39	10

Table 5: LiDAR perception extension on *Soar*.

Methods	Offline (Laptop)	*Soar* (TX2)	w/o *Soar* (TX2)
fps	10	3.67	1.73
Fusion Error (m)	0.46	0.73	1.09
Benefit Ratio (%)	42.1	37.9	15.4

driving applications on the vehicle side. Table 4 and Table 5 present the results of point cloud registration and LiDAR perception extension, respectively. The "offline" in the tables denotes the offline execution performance of our implemented applications using the original point clouds from the vehicle and infrastructure sides. Table 4 uses "success rate" and "density benefit" to present the performance of point-cloud registration and beneficial sensor data for vehicles, respectively. Table 5 leverages "fps" to show the frame rate of the perception fusion. We note that these metrics are widely adopted in various work [47, 74].

The point cloud registration requires *Soar* to transmit a large volume of raw point clouds to the vehicle, so its performance indicates the I2V communication performance. Results in Table 4 indicate that using point clouds transmitted by *Soar* with the error correction code (ECC), the vehicle can achieve a registration performance and point cloud density benefit similar to using original point clouds. This is because passive data transmission can suffer from severe packet loss without ECC. For the LiDAR perception extension, it requires only negligible results transmission while introducing a data-intensive task (i.e., 3D object detection with point cloud) to *Soar*. Table 5 shows that *Soar* achieves consistent perception fusion accuracy and the ratio of perception extension among different hardware platforms compared with the baseline. However, the baseline can only achieve 1.73 fps which incurs significant perception fusion errors (over 1 m) and benefit loss. The reason that *Soar* can achieve an acceptable fps (i.e., > 3 fps) required by the application [74] is the optimal task execution by our task management framework.

7.3 Communication Benchmark

Impact of the number of vehicles. We first investigate the impact of the number of vehicles on I2V communication. We employ a vehicle with different numbers of pseudo vehicles. The speed of vehicles is around 10 ∼ 20 km/h. Fig. 13 shows the throughput of *Soar* and two baselines with different numbers of vehicles. *Soar* and 802.11ac broadcast exhibit negligible change, while the throughput of 802.11ac unicast plummeted with more vehicles. Although 802.11ac broadcast is resistant to the vehicle number, its low throughput can not afford the large volume of data required by autonomous driving applications. On the other hand, 802.11ac unicast presents terrible scalability. In contrast, *Soar* demonstrates consistent throughputs exceeding 50 Mbps regardless of the number of vehicles. To be specific, when there are six vehicles, *Soar* achieves 3× the throughput of 802.11ac unicast.

Impact of switching overhead. We further explore the impact of switching overhead on the performance of our I2V communication. We utilize five consecutive *Soar* nodes to transfer raw LiDAR data at 30 Mbps. One vehicle drives through the nodes at different speeds. We choose 802.11ac as the baseline. Fig. 16 shows a data trace of channel switching when the vehicle moves at 20 km/h. The upper and lower lines are the throughput of 802.11ac and *Soar*, respectively. 802.11ac exhibits significant switching overhead because of its prolonged reassociation. In addition, our measurement in Fig. 4 shows that 802.11ac suffers from non-trivial switching overhead with higher vehicle speed. Such a high overhead prevents 802.11ac from being used in our I2V communication. However, *Soar* only requires adjusting the sniff channel with negligible performance degradation, which enables *Soar* to provide seamless data transmission to vehicles.

Error correction code. We evaluate the performance of the error correction code used in our passive downlink I2V broadcast. Fig. 17 shows the PDR results under various application data rates. When the data rate is low (e.g., 10 Mbps), *Soar* with error correction code can achieve less than 1% packet loss. With a large amount of data (e.g., 30 Mbps), we can still maintain over 90% of PDR. Instead, without error correction codes, the PDRs are only around 60%. The results show that *Soar* with error correction codes can achieve high-goodput and reliable transmission in different applications.

7.4 System-level Performance

We present long-term system-level performance of *Soar* in this section. First, we evaluate the long-term communication performance of our deployment by collecting throughput from each node to the sink node between Oct. 2022 and Dec. 2022. Fig. 14 shows that the average throughput consistently achieved 100 *Mbps*, except a notable performance degradation occurred on 7 Dec. 2022 due to the failure of one node. However, with our semi-fixed routing strategy (see § 4.1) nearby *Soar* nodes performed link recovery to establish new links, which allows *Soar* to still maintain around 50 *Mbps* during the transition. The error bar in Fig. 14 shows a percentile interval from 25% to 75%, which illustrates a considerable level of link instability (e.g., over 50 *Mbps*). To investigate the performance of each link of *Soar* nodes, we analyze the per-link throughput of *Soar* Cluster 2 in Oct. 2022. Fig. 15 shows that the throughput of links varies substantially, which is caused by the differences and dynamics in the physical environment of each node. Nevertheless, *Soar* can achieve over 100 *Mbps* throughput for all links.

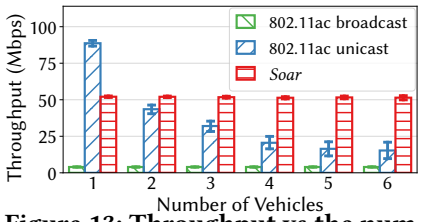


Figure 13: Throughput vs the number of vehicles.

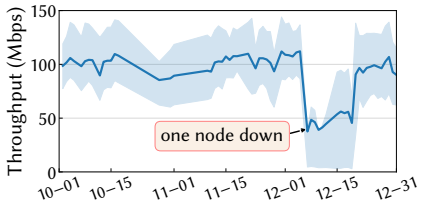


Figure 14: 3-month throughput evaluation of *Soar* in 2022.

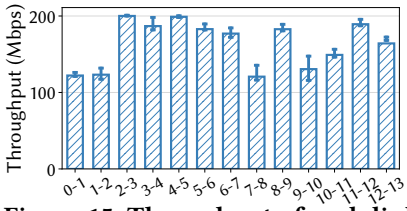


Figure 15: Throughput of each link of *Soar* on Oct. 2022.

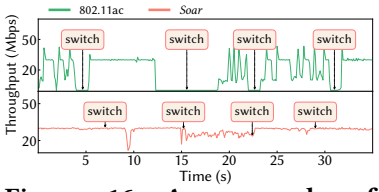


Figure 16: An example of switching over *Soar* nodes.

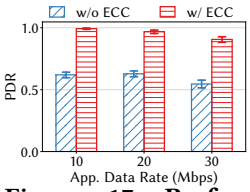


Figure 17: Performance of ECC.

It is well known that the CPU/GPU performance can degrade considerably at high temperatures [5]. We further tested the computing performance of the CPU and GPU on *Soar* over a period of three months, with an average temperature of 20 ∼ 29°*C*. We use *sysbench* [20] and *mixbench* [55] to evaluate the CPU and GPU performance, respectively. The experimental results reveal that the average execution time for a *sysbench* event on CPU ranges from 90.88 *ms* to 90.91 *ms*, indicating no significant performance degradation. The GPU performance is measured to be 20.79 *GFLOPS* on average, with a standard deviation of 0.005. Overall, the results demonstrate no significant performance fluctuations for the CPU/GPU of our platforms over several months despite varying temperatures and prolonged use.

8 DISCUSSION

Cost-effectiveness. *Soar* provides a reference design of SRI using inexpensive off-the-shelf computing and sensing components based on practical power and budget constraints of operational lampposts. *Soar*'s modular hardware design and data management framework enable cost-effective part replacement and plug-and-play functionality with powerful processors and multi-modal sensors. This design supports different components (e.g., communication, computing, etc.) to be upgraded in a cost-effective manner. In addition, *Soar* nodes can be deployed with diverse densities to enable a trade-off between the assistive services for vehicles and the deployment cost. For example, *Soar* nodes can be strategically deployed densely where they are most needed, such as busy intersections and crosswalks.

Scalability. *Soar*'s cluster-based networking and task management architecture facilitates large-scale deployments without compromising on application performance. *Soar* efficiently broadcasts data in universal formats, such as raw sensor samples or bounding boxes, which seamlessly supports a majority of current autonomous driving applications. This paradigm is inherently scalable, as it simplifies the data dissemination process and ensures that *Soar* can accommodate an expanding network of vehicles.

Security Issues. In this work, we assume that *Soar* functions under secure settings and the data transmission between infrastructure and vehicles is trusted. In real-world operations, various security measures can be integrated to enhance the security of *Soar*, such as encryption of sensor data [45], safeguarding model inference with access controls [48], and authentication protocols for communication [25].

9 CONCLUSION

This paper presents the design and deployment of *Soar*, the first end-to-end SRI system specifically designed for supporting AVs. *Soar* consists of carefully designed components for data and DL task management, I2I and I2V communication, and an integrated hardware platform, which addresses a multitude set of system and physical challenges, allows to leverage the existing operational traffic infrastructure, and hence lowers the barrier of adoption. Based on a real-world deployment of 18 *Soar* nodes on existing lampposts on campus, our evaluation shows that *Soar* can support a diverse set of autonomous driving applications, and achieve desirable real-time performance and high communication reliability.

Our experience offers key insights into the development and deployment of next-generation SRI and autonomous driving systems. *Soar* demonstrates a highly efficient yet practical instance in a fairly large design space for SRI. In the future, we will integrate emerging open communication systems based on advanced V2X technologies. In particular, an open question is how to implement high-bandwidth I2I and I2V communication in a cost-effective manner using 5G and emerging 6G cellular technologies. Lastly, we will investigate new cyber-physical security mechanisms that can protect AVs, which become increasingly critical with the prominence of smart roadside infrastructure.

ACKNOWLEDGMENTS

This work is supported in part by the Research Grants Council (RGC) of Hong Kong under General Research Fund under Grant 14222222, and the Centre for Perceptual and Interactive Intelligence (CPII) under Grant EW01610 (RP4-3).

REFERENCES

[1] 2021. Products of commsignia. https://www.commsignia.com/products/

[2] 2021. Self-driving technology: Automated Transportation Systems. https://www.tusimple.com/technology/

[3] 2022. How Many Watts Does A Street Light Use? https://kanglight.com/how-many-watts-does-a-street-light-use/.

[4] 2022. Speedtest by Ookla - The Global Broadband Speed Test. https://www.speedtest.net/.

[5] 2022. Thermal Throttling Guide (Prevent your GPU and CPU from Thermal throttling). https://www.cgdirector.com/thermal-throttling-guide/.

[6] 2023. https://www.testfeld-friedrichshafen.de/

[7] 2023. 8814au. https://github.com/morrownr/8814au.

[8] 2023. Apollo. https://apollo.auto/.

[9] 2023. The Autoware Foundation - open source for autonomous driving. https://www.autoware.org/

[10] 2023. AWS Cloud Ping. https://www.cloudping.cloud/aws.

[11] 2023. Cloud Computing Services - Amazon Web Services (AWS). https://aws.amazon.com/.

[12] 2023. Deploy AI-powered autonomous machines at scale. https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-agx-xavier/

[13] 2023. Horizon. https://www.livoxtech.com/horizon

[14] 2023. iperf3: A TCP, UDP, and SCTP network bandwidth measurement tool. https://github.com/esnet/iperf.

[15] 2023. IWR6843ISK Radar. https://www.ti.com/product/IWR6843ISK/part-details/IWR6843ISK

[16] 2023. netfilter/iptables project homepage - The netfilter.org project. https://www.netfilter.org/.

[17] 2023. Production-grade container orchestration. https://kubernetes.io/

[18] 2023. Puck lidar sensor, high-value surround Lidar. https://velodynelidar.com/products/puck/

[19] 2023. Smart mobility roadside infrastructure (art/298cp): Astri - Hong Kong Applied Science and Technology Research Institute Company Limited. https://www.astri.org/rdprojects/smart-mobility-roadside-infrastructure-art-298cp/

[20] 2023. Sysbench. https://github.com/akopytov/sysbench/.

[21] Rudolf Ahlswede, Ning Cai, S-YR Li, and Raymond W Yeung. 2000. Network information flow. *IEEE Transactions on information theory* 46, 4 (2000), 1204–1216.

[22] Ammar Mohammed Al-Jubari, Mohamed Othman, Borhanuddin Mohd Ali, and Nor Asilah Wati Abdul Hamid. 2011. TCP performance in multi-hop wireless ad hoc networks: challenges and solution. *EURASIP Journal on Wireless Communications and Networking* 2011, 1 (2011), 1–25.

[23] Waqar Anwar, Norman Franchi, and Gerhard Fettweis. 2019. Physical layer evaluation of V2X communications technologies: 5G NR-V2X, LTE-V2X, IEEE 802.11 bd, and IEEE 802.11 p. In *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*. IEEE, 1–7.

[24] Eduardo Arnold, Mehrdad Dianati, Robert de Temple, and Saber Fallah. 2020. Cooperative perception for 3D object detection in driving scenarios using infrastructure sensors. *IEEE Transactions on Intelligent Transportation Systems* (2020).

[25] Palak Bagga, Ashok Kumar Das, Mohammad Wazid, Joel JPC Rodrigues, and Youngho Park. 2020. Authentication protocols in internet of vehicles: Taxonomy, analysis, and challenges. *Ieee Access* 8 (2020), 54314–54344.

[26] Zhengwei Bai, Saswat Priyadarshi Nayak, Xuanpeng Zhao, Guoyuan Wu, Matthew J Barth, Xuewei Qi, Yongkang Liu, and Kentaro Oguchi. 2022. Cyber mobility mirror: Deep learning-based real-time 3d object perception and reconstruction using roadside lidar. *arXiv preprint arXiv:2202.13505* (2022).

[27] Soroush Bateni, Husheng Zhou, Yuankun Zhu, and Cong Liu. 2018. Predjoule: A timing-predictable energy optimization framework for deep neural networks. In *2018 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 107–118.

[28] PAUL BERRIMAN. 2019. Connecting a Smart City Future in Hong Kong with 5G.

[29] John Bicket, Daniel Aguayo, Sanjit Biswas, and Robert Morris. 2005. Architecture and evaluation of an unplanned 802.11 b mesh network. In *Proceedings of the 11th annual international conference on Mobile computing and networking*. 31–42.

[30] Car 2 Car. 2023. C-ITS: Cooperative Intelligent Transport Systems and Services. https://www.car-2-car.org/about-c-its/

[31] Julie Castermans. 2023. CVIS. http://www.cvisproject.org/

[32] Emre Cicek and Sezer Gören. 2021. Fully automated roadside parking spot detection in real time with deep learning. *Concurrency and Computation: Practice and Experience* 33, 23 (2021), e6006.

[33] Christian Cress and Alois Knoll. 2021. Intelligent Transportation Systems With The Use of External Infrastructure: A Literature Survey. *ArXiv* abs/2112.05615 (2021).

[34] Douglas SJ De Couto, Daniel Aguayo, John Bicket, and Robert Morris. 2003. A high-throughput path metric for multi-hop wireless routing. In *Proceedings of the 9th annual international conference on Mobile computing and networking*. 134–146.

[35] Thibault Degrande, Simon Van den Eynde, Frederic Vannieuwenborg, Didier Colle, and Sofie Verbrugge. 2021. C-ITS road-side unit deployment on highways with ITS road-side systems: A techno-economic approach. *IET Intelligent Transport Systems* 15, 7 (2021), 863–874.

[36] Whitfield Diffie and Martin E Hellman. 2022. New directions in cryptography. In *Democratizing Cryptography: The Work of Whitfield Diffie and Martin Hellman*. 365–390.

[37] Biyi Fang, Xiao Zeng, and Mi Zhang. 2018. Nestdnn: Resource-aware multi-tenant on-device deep learning for continuous mobile vision. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*. 115–127.

[38] Tobias Fleck, Karam Daaboul, Michael Weber, Philip Schörner, Marek Wehmer, Jens Doll, Stefan Orf, Nico Sußmann, Christian Hubschneider, Marc René Zofka, et al. 2018. Towards large scale urban traffic reference data: Smart infrastructure in the test area autonomous driving baden-württemberg. In *International Conference on Intelligent Autonomous Systems*. Springer, 964–982.

[39] Teledyne FLIR. 2022. FREE Teledyne FLIR Thermal Dataset for Algorithm Training. https://www.flir.asia/oem/adas/adas-dataset-form/.

[40] Robert Gallager. 1962. Low-density parity-check codes. *IRE Transactions on information theory* 8, 1 (1962), 21–28.

[41] Andreas Geiger, Philip Lenz, and Raquel Urtasun. 2012. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[42] Datang Gohigh. 2023. Gohigh: C-V2X. https://www.gohigh.com.cn/layout.aspx?id=95

[43] Swaminathan Gopalswamy and Sivakumar Rathinam. 2018. Infrastructure enabled autonomy: A distributed intelligence architecture for autonomous vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 986–992.

[44] Abderrahim Guerna, Salim Bitam, and Carlos T Calafate. 2022. Roadside unit deployment in internet of vehicles systems: A survey. *Sensors* 22, 9 (2022), 3190.

[45] Alma E Guerrero-Sanchez, Edgar A Rivas-Araiza, Jose Luis Gonzalez-Cordoba, Manuel Toledano-Ayala, and Andras Takacs. 2020. Blockchain mechanism and symmetric encryption in a wireless sensor network. *Sensors* 20, 10 (2020), 2798.

[46] Yuze He, Li Ma, Jiahe Cui, Zhenyu Yan, Guoliang Xing, Sen Wang, Qintao Hu, and Chen Pan. 2023. AutoMatch: Leveraging Traffic Camera to Improve Perception and Localization of Autonomous Vehicles. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems* (Boston, Massachusetts) *(SenSys '22)*. Association for Computing Machinery, New York, NY, USA, 16–30. https://doi.org/10.1145/3560905.3568519

[47] Yuze He, Li Ma, Zhehao Jiang, Yi Tang, and Guoliang Xing. 2021. VI-eye: semantic-based 3D point cloud registration for infrastructure-assisted autonomous driving. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*. 573–586.

[48] Zecheng He, Tianwei Zhang, and Ruby B Lee. 2020. Attacking and protecting data privacy in edge–cloud collaborative inference systems. *IEEE Internet of Things Journal* 8, 12 (2020), 9706–9716.

[49] Gereon Hinz, Martin Buechel, Frederik Diehl, Malte Schellmann, and Alois Knoll. 2017. Designing a far-reaching view for highway traffic scenarios with 5G-based intelligent infrastructure. In *8. Tagung Fahrerassistenz*.

[50] Tracey Ho, Ralf Koetter, Muriel Medard, David R Karger, and Michelle Effros. 2003. The benefits of coding over routing in a randomized setting. (2003).

[51] Chien-Chun Hung, Ganesh Ananthanarayanan, Peter Bodik, Leana Golubchik, Minlan Yu, Paramvir Bahl, and Matthai Philipose. 2018. Videoedge: Processing camera streams using hierarchical clusters. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 115–131.

[52] Zhehao Jiang, Neiwen Ling, Xuan Huang, Shuyao Shi, Chenhao Wu, Xiaoguang Zhao, Zhenyu Yan, and Guoliang Xing. 2023. CoEdge: A Cooperative Edge System for Distributed Real-Time Deep Learning Tasks. In *Proceedings of the 22nd International Conference on Information Processing in Sensor Networks*. 53–66.

[53] David B Johnson and David A Maltz. 1996. Dynamic source routing in ad hoc wireless networks. *Mobile computing* (1996), 153–181.

[54] Woosung Kang, Kilho Lee, Jinkyu Lee, Insik Shin, and Hoon Sung Chwa. 2021. Lalarand: Flexible layer-by-layer cpu/gpu scheduling for real-time dnn tasks. In *2021 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 329–341.

[55] Elias Konstantinidis and Yiannis Cotronis. 2017. A quantitative roofline model for GPU kernel performance estimation using micro-benchmarks and hardware metric profiling. *J. Parallel and Distrib. Comput.* 107 (2017), 37–56.

[56] Annkathrin Krämmer, Christoph Schöller, Dhiraj Gulati, Venkatnarayanan Lakshminarasimhan, Franz Kurz, Dominik Rosenbaum, Claus Lenz, and Alois Knoll. 2019. Providentia-a large-scale sensor system for the assistance of autonomous vehicles and its evaluation. *arXiv preprint arXiv:1906.06789* (2019).

[57] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. 2019. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 12697–12705.

[58] S-YR Li, Raymond W Yeung, and Ning Cai. 2003. Linear network coding. *IEEE transactions on information theory* 49, 2 (2003), 371–381.

[59] Neiwen Ling, Xuan Huang, Zhihe Zhao, Nan Guan, Zhenyu Yan, and Guoliang Xing. 2022. BlastNet: Exploiting Duo-Blocks for Cross-Processor Real-Time DNN Inference. In *Proceedings of the Twentieth ACM Conference on Embedded Networked Sensor Systems*. 91–105.

[60] Neiwen Ling, Kai Wang, Yuze He, Guoliang Xing, and Daqi Xie. 2021. Rt-mdl: Supporting real-time mixed deep learning tasks on edge platforms. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*. 1–14.

[61] Qiang Liu, Tao Han, Jiang Linda Xie, and BaekGyu Kim. 2021. Livemap: Real-time dynamic map in automotive edge computing. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 1–10.

[62] M Luby, A Shokrollahi, M Watson, T Stockhammer, and L Minder. 2011. RaptorQ forward error correction scheme for object delivery–rfc 6330. *IETF Request For Comments* (2011).

[63] Qian Luo, Yurui Cao, Jiajia Liu, and Abderrahim Benslimane. 2019. Localization and navigation in autonomous driving: Threats and countermeasures. *IEEE Wireless Communications* 26, 4 (2019), 38–45.

[64] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. 2022. Robot Operating System 2: Design, architecture, and uses in the wild. *Science Robotics* 7, 66 (2022), eabm6074. https://doi.org/10.1126/scirobotics.abm6074

[65] Gaurang Naik, Biplav Choudhury, and Jung-Min Park. 2019. IEEE 802.11 bd & 5G NR V2X: Evolution of radio access technologies for V2X communications. *IEEE access* 7 (2019), 70169–70184.

[66] NVIDIA. 2023. Jetson TX2 Module. https://developer.nvidia.com/embedded/jetson-tx2.

[67] Risto Ojala, Jari Vepsäläinen, Jussi Hanhirova, Vesa Hirvisalo, and Kari Tammi. 2019. Novel convolutional neural network-based roadside unit for accurate pedestrian localization. *IEEE Transactions on Intelligent Transportation Systems* 21, 9 (2019), 3756–3765.

[68] Perbadanan. 2002. . https://www.ppj.gov.my/storage/putrajaya02/239/239.pdf

[69] Charles Perkins, Elizabeth Belding-Royer, and Samir Das. 2003. *Ad hoc on-demand distance vector (AODV) routing*. Technical Report.

[70] Charles E Perkins and Pravin Bhagwat. 1994. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. *ACM SIGCOMM computer communication review* 24, 4 (1994), 234–244.

[71] KubeEdge Project. 2023. KubeEdge. https://kubeedge.io/en/.

[72] Hang Qiu, Fawad Ahmad, Fan Bai, Marco Gruteser, and Ramesh Govindan. 2018. Avr: Augmented vehicular reality. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*. 81–95.

[73] Siegfried Seebacher, Bernd Datler, Jacqueline Erhart, Gerhard Greiner, Manfred Harrer, Peter Hrassnig, Arnold Präsent, Christian Schwarzl, and Martin Ullrich. 2019. Infrastructure data fusion for validation and future enhancements of autonomous vehicles' perception on Austrian motorways. In *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE)*. IEEE, 1–7.

[74] Shuyao Shi, Jiahe Cui, Zhehao Jiang, Zhenyu Yan, Guoliang Xing, Jianwei Niu, and Zhenchao Ouyang. 2022. VIPS: real-time perception fusion for infrastructure-assisted autonomous driving. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*. 133–146.

[75] Amin Shokrollahi. 2006. Raptor codes. *IEEE transactions on information theory* 52, 6 (2006), 2551–2567.

[76] Cristiano M Silva, Andre LL Aquino, and Wagner Meira Jr. 2015. Deployment of roadside units based on partial mobility information. *Computer Communications* 60 (2015), 28–39.

[77] Mingxing Tan and Quoc Le. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*. PMLR, 6105–6114.

[78] Tuyen X Tran, Abolfazl Hajisami, Parul Pandey, and Dario Pompili. 2017. Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges. *IEEE Communications Magazine* 55, 4 (2017), 54–61.

[79] Manabu Tsukada, Takaharu Oi, Masahiro Kitazawa, and Hiroshi Esaki. 2020. Networked Roadside Perception Units for Autonomous Driving. *Sensors* 20, 18 (2020). https://doi.org/10.3390/s20185320

[80] Wikipedia. 2023. Packet analyzer. https://en.wikipedia.org/wiki/Packet_analyzer

[81] Wikipedia. 2023. Packet injection. https://en.wikipedia.org/wiki/Packet_injection

[82] Jianqing Wu, Hao Xu, and Jianying Zheng. 2017. Automatic background filtering and lane identification with roadside LiDAR data. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 1–6.

[83] Shenghao Yang and Raymond W Yeung. 2014. Batched sparse codes. *IEEE Transactions on Information Theory* 60, 9 (2014), 5322–5346.

[84] Shenghao Yang, Raymond W Yeung, Jay HF Cheung, and Hoover HF Yin. 2014. BATS: Network coding in action. In *2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 1204–1211.

[85] Juheon Yi and Youngki Lee. 2020. Heimdall: mobile GPU coordination platform for augmented reality applications. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*. 1–14.

[86] Xiao Zeng, Biyi Fang, Haichen Shen, and Mi Zhang. 2020. Distream: scaling live video analytics with workload-adaptive distributed edge intelligence. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*. 409–421.

[87] Huazi Zhang, Kairan Sun, Qiuyuan Huang, Yonggang Wen, and Dapeng Wu. 2016. FUN coding: Design and analysis. *IEEE/ACM Transactions on Networking* 24, 6 (2016), 3340–3353.

[88] Xumiao Zhang, Anlan Zhang, Jiachen Sun, Xiao Zhu, Y Ethan Guo, Feng Qian, and Z Morley Mao. 2021. Emp: Edge-assisted multi-vehicle perception. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*. 545–558.

[89] Junxuan Zhao, Hao Xu, Hongchao Liu, Jianqing Wu, Yichen Zheng, and Dayong Wu. 2019. Detection and tracking of pedestrians and vehicles using roadside LiDAR sensors. *Transportation research part C: emerging technologies* 100 (2019), 68–87.

[90] Xiangmo Zhao, Kenan Mu, Fei Hui, and Christian Prehofer. 2017. A cooperative vehicle-infrastructure based urban driving environment perception method using a DS theory-based credibility map. *Optik* 138 (2017), 407–415.

[91] Zhiheng Zhou, Congduan Li, Shenghao Yang, and Xuan Guang. 2019. Practical inner codes for BATS codes in multi-hop wireless networks. *IEEE Transactions on Vehicular Technology* 68, 3 (2019), 2751–2762.

[92] Zhengxia Zou, Rusheng Zhang, Shengyin Shen, Gaurav Pandey, Punarjay Chakravarty, Armin Parchami, and Henry X Liu. 2022. Real-time full-stack traffic scene perception for autonomous driving with roadside cameras. In *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 890–896.