A Real-Time Implementation of an Edge-Assisted Collision Warning System for Urban Intersections

A Real-time Implementation of an Edge-Assisted Collision Warning System for Urban Intersections

Christian Vitale, Panagiotis Kardaras, Panayiotis Kolios, Carla Fabiana Chiasserini, and Georgios Ellinas

***Abstract*— Communication-based collision warning systems (CWSs) play a pivotal role in enhancing safety, particularly during adverse weather conditions or when obstacles obstruct drivers' and sensors' visibility. In this study, we implement and validate a practical and scalable CWS tailored for urban intersections, incorporating machine learning-based trajectory and collision predictions, Cooperative Awareness Message dissemination, and Decentralized Environmental Notification Message alerts. The validation process entails testing the system on a cellular testbed, exploiting Open Air Interface, and leveraging realistic traffic data to closely replicate dense urban scenarios. Through extensive experimental testing, our findings demonstrate the system's efficacy in proactively identifying dangerous situations well in advance, even within densely populated urban environments, thereby enabling timely evasive maneuvers.**

I. Introduction

Road safety remains a significant concern for the automotive sector, as the number of motor vehicle traffic crashes shows no signs of decreasing, with 42,795 fatalities recorded in 2022 in the United States alone [1]. To confront this pressing issue, the automotive industry has taken proactive measures, initially by outfitting new vehicles with advanced active safety systems and more recently by integrating communication technologies at the core of its safety strategies [2]. Notably, vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) technologies empower vehicles to obtain real-time information from nearby vehicles and smart city sensors, significantly enhancing their situational awareness. Moreover, unlike vision- or radar-based systems, the data transmitted through V2V and V2I networks exhibit resilience to adverse weather conditions and non-line-of-sight scenarios, thus consistently improving achievable road safety [3]. This capability is largely facilitated by the adoption of 5G connectivity onboard vehicles, characterized by pervasive coverage and low latency, which serve as the cornerstone for efficient cellular vehicle-to-anything (C-V2X) communications [4]. Consequently, it comes as no surprise that collision warning systems (CWSs) reliant on wireless communication technologies have emerged as a widely-researched topic.

Due to the tight latency constraints of CWSs in dense urban scenarios, edge-assisted solutions often prioritize minimal complexity over safety, relying on: (i) predictions of vehicle trajectories and driver intentions, and (ii) collision prediction and warning transmission. To mitigate computational complexity, one of these components undergoes significant simplification or relies heavily on assumptions that hardly hold in real-world scenarios.

For example, modeling the probability of vehicle collision is a common approach. [5] calculates a range of trajectories based on assumptions about the probability of applied deceleration/acceleration and [6] generates bounding boxes encompassing vehicles with a certain probability. In both cases, collision detection occurs when the probability of vehicles' trajectories nearing each other surpasses a set threshold. Although theoretically valuable, the assumptions underpinning such approaches are strong and challenging to fulfill in real-world application scenarios.

Another set of approaches forecasts driver intentions and, based on them, detects collisions. For instance, [7] employs dynamic Bayesian networks to ascertain whether the driver's trajectory controls (steering and acceleration/deceleration) align with known hazardous maneuvers, while [8] uses back-propagation neural networks (NNs) to anticipate the actions of the driver and, based on these, predict future vehicle locations. In both cases, when two vehicles exhibit a dangerous pattern, the expected time to collision (TTC) is computed and used to determine whether or not to generate an alarm. While these approaches rely on fewer assumptions, they still require quantizing potential human maneuvers, resulting in significant approximations of the risk posed by vehicles, especially in scenarios where many maneuvers are possible, i.e., at intersections.

Finally, collisions may be predicted using various trajectory prediction techniques. In this context, [9] utilizes constant turn and velocity motion models to anticipate vehicles' future trajectories, while [10] and [11] use third and fourth-degree polynomials. Given these trajectories, TTC is used to determine if a collision is imminent and, if so, trigger an alarm. However, such solutions require making significant approximations in challenging environments like urban intersections [12], resulting in CWS solutions that fail to meet safety standards. Unlike the above approaches, [13] proposes an edge-assisted CWS utilizing data-driven precise trajectory predictions, even at intersections, through long short-term memory (LSTM) recurrent NNs. Subsequently, a random

C. Vitale, P. Kardaras, and G. Ellinas are with the KIOS Research and Innovation Center of Excellence (KIOS CoE) and the Department of Electrical and Computer Engineering, University of Cyprus, Nicosia, Cyprus. P. Kolios is with the KIOS CoE and the Department of Computer Science, University of Cyprus, Nicosia, Cyprus. (e-mail:{vitale.christian, kardaras.panagiotis, pkolios, gellinas}@ucy.ac.cy). C.F. Chiasserini is with CARS@Polito and Politecnico di Torino, Torino, Italy (e-mail:{carla.chiasserini}@polito.it).

This work was supported by the EU's Horizon 2020 research and innovation programme under grant agreements No 739551 (KIOS CoE - TEAMING) and No. 101069688 (CONNECT), and from the Republic of Cyprus through the Deputy Ministry of Research, Innovation and Digital Policy.

forest classifier (RFC) identifies hazardous situations based on experience gained from monitoring the intersection of interest. Nevertheless, [13] lacks realistic testing to determine practical feasibility. This is generally the case for all existing methods, as they rely solely on offline simulations (using SUMO or MATLAB) or simplified emulations with a small number of vehicles to assess the effectiveness of the proposed CWSs [14], [15].

To overcome the limitations of current approaches, this work introduces, for the first time, the application of a CWS solution in dense urban scenarios through a realistic experimental setup. The proposed implementation builds upon [13] as a centralized solution. It leverages hardware-in-the-loop cellular communications, based on Open Air Interface (OAI), to emulate V2I packet exchanges. Specifically, standard Cooperative Awareness Messages (CAMs) are employed for vehicle state dissemination, while Decentralized Environmental Notification Messages (DENMs) are used to alert vehicles in danger. CAMs from multiple vehicles are transmitted in the system through a software emulator attached to the OAI cellular network. These CAMs, derived from real traffic traces, simulate dense traffic scenarios, potentially risky driver decisions, and sensor measurement errors. The proposed validation demonstrates the system's ability to preemptively identify hazardous situations well in advance, enabling practical evasive maneuvers.

Overall, the contributions of this work are as follows:

- It proposes an experimental setup capable of testing automotive safety applications, such as CWS solutions, accounting for realistic communication, computational latency, and vehicular traffic scenarios.
- For the first time, it implements and validates the performance of a CWS solution in a dense urban scenario.
- Exploiting the proposed experimental setup, it showcases the capabilities of the implemented CWS solution to timely predict dangerous situations, even in the presence of realistic application latencies and vehicle on-board sensor measurement errors.

The remainder of the paper is structured as follows. Section II summarizes the implemented CWS solution. Section III showcases the experimental set-up and delves into the implementation details of the CWS solution. Section IV presents the validation of the CWS solution in a dense urban scenario and its detailed performance evaluation. Finally, Sec. V concludes the paper.

II. The CWS Solution

In this work, we implement a state-of-the-art CWS [13], comprising two main components: (i) LSTM trajectory prediction and uncertainty estimation (Sec. II-A), and (ii) RFC-based identification of potential collisions using predicted vehicle movements (Sec. II-B). An overview of the approach is given in Fig. 1, while its implementation is detailed in Sec. III.

A. Vehicle Trajectory Prediction and Uncertainty

Trajectory prediction involves forecasting future vehicle paths based on past trajectories and relevant data. In this work, we use LSTMs, as they have demonstrated good performance in sequence-to-sequence prediction tasks, such as trajectory prediction [12]. Nevertheless, LSTMs estimate future vehicle positions over multiple steps with a degree of uncertainty. Point-estimates are inadequate for collision avoidance, as they may lead to erroneous predictions (causing driver frustration) or, even worse, underestimation of collision risks (leading to non-timely issued alerts, resulting in collisions). Thus, the proposed technique builds both trajectory predictions and prediction intervals to measure the certainty level of trajectory estimates, capturing the existence of multimodal future trajectories [16]. This approach reduces errors by minimizing false collision predictions when average trajectory predictions suggest potential crashes but the prediction is affected by high uncertainty. Also, it does not underestimate threats when seemingly safe trajectory predictions are associated with large uncertainty.

1) Least Squares Approach for Predicting Trajectories - Mean Response: To approximate the relationship between inputs and vehicles' trajectories prediction, the LSTM is trained to minimize the mean squared error (MSE) between predictions and future actual vehicles' locations:

$$L_{mse} = \frac{1}{NL}\sum_{i=1}^{N}\sum_{j=1}^{L} ||\mathbf{y}_{i,t+j} - \hat{\mathbf{y}}_{i,t+j}||_2^2. \tag{1}$$

In (1), N is the number of training sequences, L are the prediction time-steps, t is the current time, vector $\mathbf{y}_{i,t+j}\in\mathbb{R}^2$ describes vehicle i's true location at $t+j$, and $\hat{\mathbf{y}}_{i,t+j}\in\mathbb{R}^2$ is the estimated vector inferred using vector $\mathbf{x}_i\in\mathbb{R}^d$ vehicle i's input data by the trained least squares LSTM model.

2) Deep Quantile Regression for Estimating Uncertainty - Lower and Upper Boundaries: The accuracy of trajectory predictions is obtained by measuring the associated uncertainty (i.e., stemming from potential multimodal future trajectories), and accounting for it when predicting collisions. Uncertainty estimation relies on constructing prediction intervals, defining a range of values within which a future observation is expected to fall with a certain probability. To achieve this, an LSTM is trained to simultaneously estimate a lower quantile, l, and an upper quantile, u. Obtaining reliable prediction intervals is achieved by minimizing during training the following asymmetrically weighted sum of absolute errors [17]:

$$L_q=\frac{1}{2NL}\sum_{i=1}^{N}\sum_{j=1}^{L}\rho_l\big(\mathbf{y}_{i,t+j}-\hat{\mathbf{y}}^{l}_{i,t+j}\big)+\rho_u\big(\mathbf{y}_{i,t+j}-\hat{\mathbf{y}}^{u}_{i,t+j}\big), \tag{2}$$

where ρ_q denotes (for both u and l), the following function:

$$\rho_q(z) = \begin{cases} qz, & \text{if } z \geq 0, \\ (q-1)z, & \text{if } z < 0 \end{cases}, \tag{3}$$

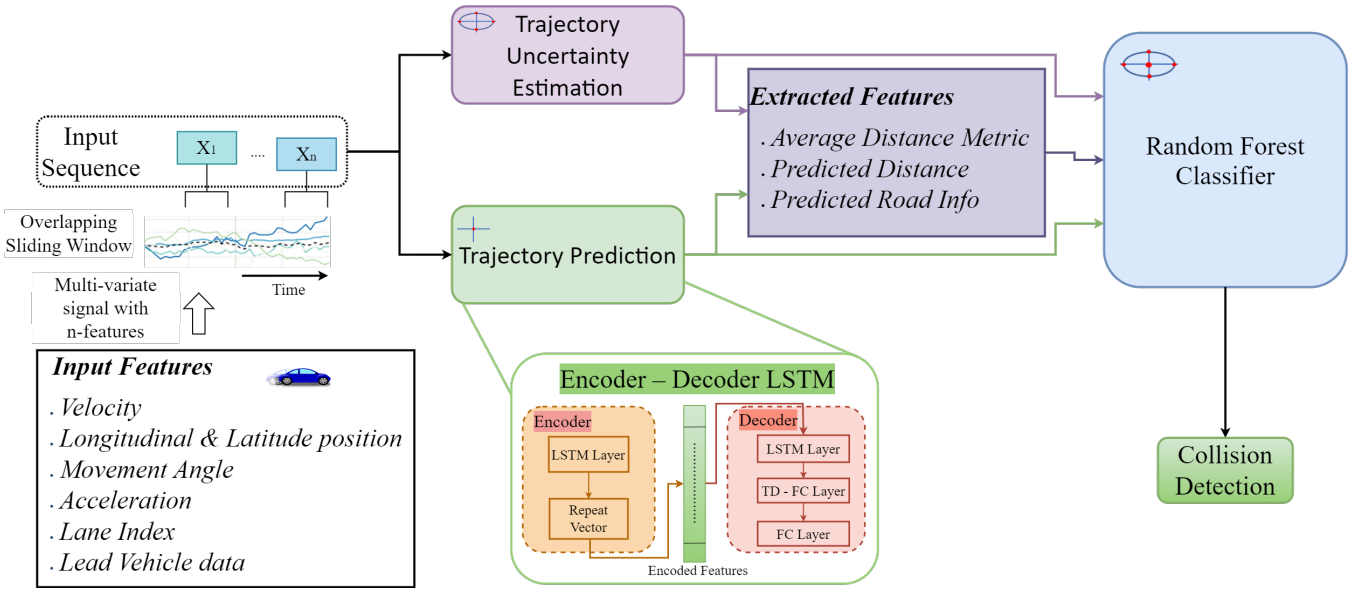

Fig. 1. Proposed network-assisted collision prediction methodology.

with $\mathbf{y}_{i,t+j} \in \mathbb{R}^2$ denoting the actual vehicle location at $t+j$ and $\hat{\mathbf{y}}^q_{i,t+j}$ signifying the estimated q-quantile. Separate LSTM models are trained to provide longitude and latitude upper and lower estimates for the future locations of the vehicle.

3) Models Inputs: Both trajectory predictions and uncertainty estimation use the same input information. Past data observations are sampled to create a time series, with T=30 samples of past data, each spaced τ=0.1 s apart (τ represents the time scale of sampling time instants). The prediction horizon is also set at L=30 samples. The inputs encompass vehicle's latitude and longitude positions (x, y), yaw angle (ϕ), velocity (v), acceleration (a), traveled lane ID (l), location (x^p, y^p) and velocity (v^p) of the preceding vehicle. The rationale behind these inputs stems from the high correlation observed at intersections between the future driving controls applied by a vehicle and the present driving controls applied by its preceding vehicle [12].

B. ML-aided Uncertainty-aware Collision Prediction

To determine if two vehicles are on a collision course, CWS utilizes a trained RFC [18] that classifies inputs as dangerous/non-dangerous (v=1/0). During training, the RFC constructs multiple decision trees that recursively split input features into subsets at each branch (using the Gini Index [18] and training data collected from the intersection of interest), ending at leaf nodes where the final classification occurs. This training data is labeled based on what is considered dangerous, and leaf nodes are reached when inputs fall within specific feature ranges. The inputs for the RFC are directly derived from the trajectory predictions and prediction intervals obtained with LSTM for the vehicles under test, aligning with those illustrated in [13]. Thus, leaf nodes associated with collisions correspond to trajectory and uncertainty predictions similar to hazards observed when monitoring the intersection.

III. Proposed System Implementation

This section describes the real-time testbed we developed to test our proposed ML-aided CWS, characterized by low cost (utilizing commercial off-the-shelf (COTS) hardware), scalability, robustness, and prompt risk detection for drivers. The testbed, depicted in Fig. 2, consists of three main components: (i) the OAI cellular network (Sec. III-B), (ii) the user equipment (UE) acting as the intersection emulator (IE), which transmits the CAMs from the vehicles crossing the intersection (Sec. III-C), and (iii) the virtual machines (VMs) implementing the CWS (Sec. III-D). Further, Sec. III-A outlines how these components interact.

A. Testbed Components Interaction

In the developed testbed, a single UE connects to the OAI cellular network and mimics the communication traffic expected in a dense urban intersection. Leveraging open-source OAI software for both the UE and the cellular network, the UE periodically transmits CAMs for each vehicle through the OAI cellular network. These CAMs contain vehicle state information and reach a real-time database, the CAM DB, which decodes them and provides them to the edge-assisted CWS. Further, the vehicle machine learning (VML) application acts as the core intelligence of the CWS, determining whether two vehicles are or not on a collision course. The VML interface (VML INT) acts as the VML application's front-end, fetching vehicle state information from the CAM DB at regular intervals. When the VML predicts a hazardous situation, the VML INT triggers the DENM transmitter (DENM TX) to issue an alert. This modular implementation allows for seamless integration of additional safety applications alongside the VML.

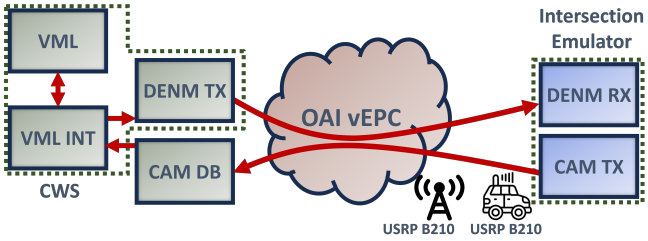

Fig. 2. Implemented testbed architecture.

B. 5G-like Connectivity

The system builds on OAI [19], an open-source implementation of both radio access network (RAN) and core network elements of a cellular network, focusing on 4G/5G compliant technology. This network comprises a software-based base station (operating on a dedicated host and equipped with a USRP B210 RF board to ensure real-time functionality) and a core network, specifically the OAI software virtual evolved packet core (OAI vEPC). The compact and portable vEPC implementation of OAI enables a reduction in end-to-end latency compared to cloud-based solutions, facilitating efficient and rapid communication between vehicles and the edge network, hence adhering to the strict latency constraints of vehicular safety applications such as the CWS.

In this setup, the CAM DB and the CWS applications operate as VMs directly on the KVM hypervisor [20] of two hosts connected to the vEPC. To enable vehicle transmissions to the CAM DB, the IE transmits CAMs to the fixed IP address of the CAM DB. This setup mirrors the functionality of a multi-access edge computing (MEC)-enabled 5G network, which utilizes specific redirection rules (via traffic filters identifying particular flows distinguished by service IP address-port-protocol tuples) to direct traffic to the relevant MEC application rather than routing it to the Internet, thereby keeping the traffic "local".

C. The Urban Intersection Emulator

As previously mentioned, in our testbed, a UE simulates the data traffic of an urban intersection. This software, i.e., the IE operates as a standalone C++ application running on a dedicated host connected to a USRP B210 RF board. The IE utilizes a standard OAI UE implementation to connect wirelessly to the OAI cellular network, taking a CSV file as input, with each line representing a single CAM (including location, yaw, velocity, acceleration, and yaw rate information). Additionally, the CSV file contains a timestamp indicating when the CAM is expected to be transmitted. Thanks to a multi-threading implementation, the IE ensures adherence to the expected transmission time. For each line in the CSV file, the IE creates a standardized structure using the open-source CAM compiler ASN1 [21] and initiates a new thread responsible for CAM creation. Once prepared, the IE transmits the CAMs via a UDP socket to the CAM DB and logs the transmission time. Thus, the IE can transmit CAMs from multiple vehicles, utilizing a frequency within the range specified by CAM standards (i.e., every 100 ms for each vehicle, resulting in a total of C CAMs every 100 ms, where C is the number of emulated vehicles).

In the event of a predicted collision, the CWS dispatches a DENM to the vehicles involved, i.e., it transmits DENMs to the IE. This functionality is facilitated by the CAM DB, which stores the IP address of the transmitting UE. The IE awaits DENMs on a specific port and upon reception, decodes the DENM transmission and records the reception time. Thanks to the identification of the CAM triggering the DENM in the CWS, this process enables the computation of end-to-end latency statistics.

D. The Automotive Safety Application

The implemented edge-assisted collision avoidance application is split into (i) the CAM DB, which receives and decodes the CAMs from the vehicles, and (ii) the CWS, which predicts if two vehicles are on a collision course.

All CAMs generated by the IE pass through the OAI vEPC before reaching the CAM DB. Operating as an autonomous C++ application running on a dedicated host, the CAM DB implementation is in line with that of an independent third-party entity, such as a transportation authority, offering support to various intelligent transportation system (ITS) applications. CAM messages from the IE are received by the CAM DB via a UDP socket, where they undergo decoding and extraction of vehicle data. Subsequently, this data is stored in RAM to ensure prompt availability for information processing when the CWS queries the CAM DB for new vehicle data.

The CWS is tasked with predicting potential collisions at the monitored intersection. The CWS comprises VML INT, VML, and DENM TX. The VML INT (a C++ application) queries the CAM DB every 5 ms via a dedicated TCP connection, striking an optimal balance between the delay introduced by periodic querying the CAM DB and the computational load. CAMs are provided by the CAM DB in aggregate form (JSON format). The VML INT interprets the CAM DB's response and prepares the data, if any, for transmission to the VML component. Due to the standardized format of transmitted CAMs, which do not permit the inclusion of all necessary information, certain inputs required by the LSTM models are absent from the CAMs (i.e., the lane ID of the ego vehicle and the location and speed of the lead vehicle). VML INT reconstructs the missing information leveraging a comprehensive view derived by processing all CAMs at the intersection and projecting all vehicle locations onto the intersection map. Again, the VML INT prepares the data of the processed CAMs in RAM for when the VML application queries new vehicle data.

The VML (a Python application) implements the CWS, effectively addressing real implementation challenges i.e., packet losses and computational latency. First, it acquires the vehicles' data in batches from the VML INT, and then prepares the input data to the LSTM models. Employing a sliding window approach, the VML systematically stores the data inputs of the last L=30 CAMs for each vehicle.

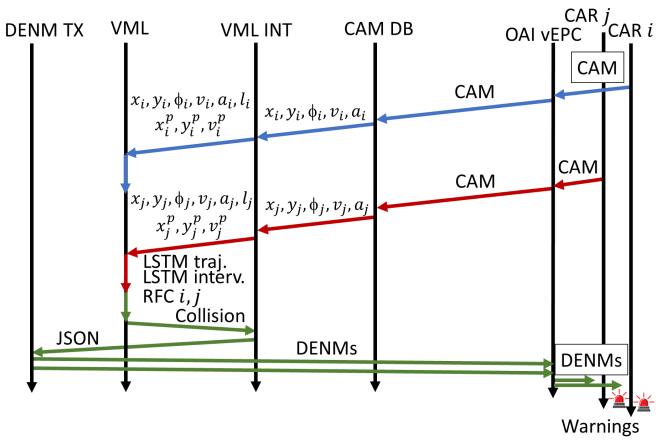

Fig. 3. Testbed components' interactions upon collision detection.

Since the VML INT communicates the timestamp of each received CAM, and thanks to the periodic nature of CAM transmissions, the VML easily identifies CAMs' losses. In the event of such losses, the VML duplicates the last received input data, ensuring the coherence of the input data. CAM timestamps are also used to prevent reshuffling issues in case of computational latency and the presence of multiple CAMs belonging to the same vehicle in the batch received by the VML. For each batch of received CAMs, the VML predicts the trajectory of each vehicle in the batch and the uncertainty associated with such prediction using LSTMs. Such computations are executed if the CWS received at least L=30 CAMs from the vehicle of interest, ensuring that the historical data is robust enough to generate meaningful predictions. To optimize the computation time of the inference stage, testing is performed using multi-threading and the obtained LSTM models are converted into their ONNX format [22]. Finally, RFC classification uses the predicted trajectories and prediction intervals to assess whether a pair of vehicles in the intersection will occupy positions in the future that may pose a risk. To enhance system efficiency and reduce computational load, the RFC model is converted to ONNX format, and vehicle pairs are examined only if: (i) the time between the last received CAMs from the two vehicles is within 300 ms; (ii) vehicles are not approaching the intersection from the same road, and (iii) the future location of the vehicles is within 50 m of the center of the intersection.

For each pair of vehicles on a collision course, the VML informs the VML INT, which in turns generate a JSON with pertinent details, including IP addresses, and sends it to the DENM TX (a C++ application). Upon receiving the JSON, the DENM TX prepares and transmits unicast DENM messages. This involves building the DENM structure, integrating JSON information, ensuring standard compliance, and encoding the DENM into a byte array. Subsequently, the DENM is transmitted via UDP socket through the OAI cellular network to the vehicles, i.e., to the IE. Figure 3 provides a summary of the interactions among testbed components when a collision is detected.

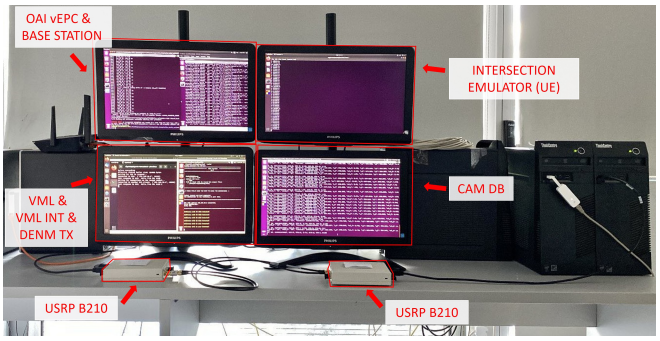

Fig. 4. The experimental testbed setup.

IV. Performance Results

In this section, the scenario selected for testing the implemented CWS framework is presented (Sec. IV-A), followed by the performance of the approach, both in terms of end-to-end latency (Sec. IV-B) and the ability of detecting a collision in time (Sec. IV-C).

A. Testbed Setup and Testing Scenario.

The testbed used for implementing the CWS consists of four hosts. A PC, equipped with a 8-cores processor running at 3.2 GHz and 8 GB of RAM and connected to a USRP B210 RF board, emulates the radio interface of the UE hosting the IE application. A second identical PC emulates the OAI radio/core cellular network and is also linked to a USRP B210 RF board. These two boards are connected via SMA cables and 30 dB attenuators. Additionally, a third identical PC hosting the CAM DB and a fourth hosting the CWS (utilizing a 12-cores processor at 3 GHz and 16 GB of RAM) are connected to the OAI cellular network via a switch. Figure 4 illustrates the configuration of the experimental setup.

To evaluate the proposed methodology, synthetic mobility data derived from validated traffic scenarios within the SUMO emulator are used, accurately portraying real-world traffic dynamics. Specifically, mobility data is obtained from a simulation that closely follows an unregulated four-way intersection in Luxembourg, featuring roads with three lanes [23]. Data extraction is conducted through SUMO's traffic control interface (TraCI) library, capturing vehicle movements between 6-11 am, encompassing the morning rush hour, and resulting in a dataset of 6,132 vehicles. Notably, all vehicles are instructed to behave aggressively, leading to 197 collisions.

For the simulation at hand, the TraCI library facilitates the acquisition of all necessary inputs for the LSTM model at a sampling interval of 100 ms, with the vehicles' location referring to the center of their front bumpers. To obtain realistic CAM transmissions, zero-mean Gaussian iid sensor measurement errors are added to the TraCI outputs. Specifically, the standard deviation of the errors are: $\sigma_{x/y} = 0.2$ m; $\sigma_\phi = 0.1^o$; $\sigma_v = 0.1$ m/s; and $\sigma_a = 0.33$ m/s^2 [24].

The final 20 minutes of the acquired data trace are allocated for testing purposes, while the remainder of the

dataset is utilized for training and validating both the LSTM and RFC models, employing an $80/20\%$ split ratio. Within the set of vehicle collision pairs, the testing subset for the Luxembourg intersection encompasses 4 collision pairs. It should be noted that extensive trials have been conducted on the hyperparameters of the LSTMs and the RFC, with the selected ones demonstrating the best performance.

B. End-to-end Delay.

To evaluate the system's ability to deliver prompt and timely notifications to drivers, the end-to-end delay (EED) at the application level is calculated (determined solely for CAMs that trigger an alarm and defined as the interval between a vehicle's transmission of a CAM and the reception of the DENM triggered by that CAM). EED encompasses both network delays and processing times. Network delays include the time required for a CAM to reach the CAM DB and the time taken for the DENM to reach the vehicle that sent the CAM. Processing times include CAM decoding, collision prediction, and DENM transmission (Fig. 3). As the CWS processes vehicle data in batches, the processing time in the EED calculation includes the processing time for all CAMs within the same batch and the time CAMs spent at different entities awaiting analysis of the previous batch.

Fig. 5 depicts the experimental CDF of the EED, derived from five different runs of the same simulation of the testing scenario, encompassing both true positive (TP) and false positive (FP) alarms. On average, 186 alarms are raised during testing, primarily due to the four collisions present in the trace (accounting for 67.5% of the alarms). Despite using only COTS hardware for the CWS solution, 98.17% of the alarms raised experience an EED of 400 ms and below. Considering that the implemented CWS utilizes trajectory predictions and uncertainty estimations with a 3-second horizon, i.e., it is designed to predict collisions 3 seconds ahead of time, the attained EED ensures that alarms promptly reach vehicles, enabling drivers to react to imminent danger even in dense urban intersections.

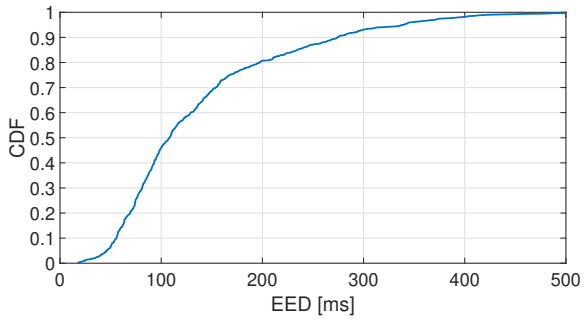

Fig. 5. CDF of the end-to-end delay of the developed CWS implementation.

Given that the CWS implementation leverages edge-assisted low-latency network communication, most of the EED shown in Fig. 5 is due to computational latency, specifically the VML's processing of the LSTM and RFC models. Fig. 6 illustrates the computation time of the VML for each batch of analyzed CAMs and the number of CAMs within each batch (which correlates closely with the number of vehicles passing through the intersection) for one of the runs performed (similar results are obtained for all runs). As expected, the VML's computation time is directly proportional to the number of CAMs in a batch and, thus, to the number of vehicles traversing the intersection. Specifically, when the number of CAMs received from the monitored intersection is 10 or less, the VML computation time is below 200 ms. When the number of CAMs grows, the VML must consider a larger number of trajectory predictions and a larger number of possibly colliding pairs, hence it requires a larger computation time (with a peak of 600 ms for a batch including 48 CAMs). Fig. 6 also illustrates the moments where collisions happen in the testing dataset. Interestingly, collisions are reported off-peak; this is the case as peaks in communication traffic correspond also to peaks in vehicular traffic, with reduced average speed and lower likelihood of collision among vehicles, even in the presence of aggressive driving behavior.

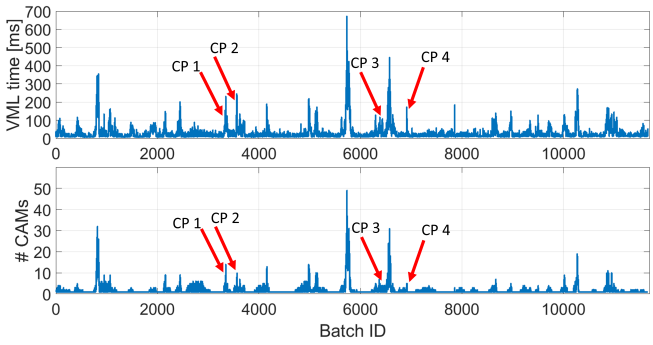

Fig. 6. Per-batch VML computational latency (top), number of CAMs per batch (bottom), and timestamps indicating collisions (marked by arrows and with the acronyms CP 1/2/3/4).

C. Collision Prediction Performance.

Given prior knowledge of vehicle collisions, three performance metrics can be computed: (i) the number of alarms raised for actual collisions, i.e., TPs, (ii) the number of collisions for which no alarm was raised, i.e., True Negatives (TNs), and (iii) the number of alarms raised for vehicles that are not involved in a collision, i.e., false positives (FP). In the 5 runs performed with our testbed, all collisions were correctly predicted, resulting in 4 TPs and 0 TNs in all runs. Notably, the number of FPs was consistently 8, referring to the same 8 pairs of vehicles. For 7 of the 8 FPs, the corresponding vehicles experience minimum distances among the chassis of the involved vehicles ranging between 10 cm and 3.26 m. A single FP involved vehicles whose minimum distance was 8.09 meters. In this case, both vehicles approached the intersection at around 50 km/h with colliding trajectories. Just after entering the intersection, one of the vehicles turned right, avoiding the danger without decelerating. Thus, for all 8 false positives, the corresponding vehicles were involved in potentially dangerous situations that narrowly avoided accidents.

The ability to predict collisions timely is assessed by evaluating whether or not the involved vehicles had enough time to brake before impact. To this end, the vehicles' positions and speeds at the reception of the first DENM are recorded. Then, focusing on human-driven vehicles, the time needed for the vehicle's human-to-machine interface (HMI) to process the warning message (set at $400\,\text{ms}$) and the human reaction time (set at $1\,\text{s}$) are factored in. Finally, using a constant deceleration rate (set at $4.5\,\text{m/s}^2$), the position of the vehicles when stopping can be calculated to determine if the collision was predicted in time. Table I summarizes both the average time available for drivers to brake and the distance between the chassis of the vehicles when stopping, after applying brakes (averaged over 5 simulation runs). The implemented CWS can predict collisions in a timely manner, and all collisions are averted. Notably, this is true even when considering that in all collisions, apart from the third colliding pair, at least one of the vehicles is traveling with a speed of $50\,\text{km/h}$ or more. Hence, the presented results confirm that the implemented low-cost CWS solution is scalable, robust, and able to issue prompt warnings to drivers experiencing dangerous situations.

TABLE I

SAFETY METRICS OBTAINED ON COLLISIONS BY THE IMPLEMENTED CWS

	Time to Collision Mean ± Std [s]	Min. Distance Mean ± Std [m]
CP 1	2.935 ± 0.038	1.36 ± 0.06
CP 2	3.751 ± 0.142	10.7 ± 3.03
CP 3	3.197 ± 0.747	4.12 ± 1.27
CP 4	2.643 ± 0.032	2.82 ± 0.14

V. CONCLUSION

The possibility of developing low-latency applications that could revolutionize advanced road safety has been opened up by the deployment of 5G and edge computing platforms. Intersection collision avoidance stands out as an exemplary application of this technology. In this work, a CWS is implemented and tested under realistic conditions. Through hardware-in-the-loop validations, we fine-tuned deployment parameters and demonstrated the effectiveness of the approach in preventing collisions and safely halting vehicles well before potential crashes even in dense urban intersections.

Future research directions include integrating small (scaled at 1:10) connected autonomous vehicles into the testbed to explore the capabilities of 5G networks to support autonomous intersection management algorithms.

REFERENCES

[1] NHTSA, "NHTSA estimates for 2022 show roadway fatalities remain flat after two years of dramatic increases," Tech. Rep., 2023. [Online]. Available: https://www.nhtsa.gov/press-releases/traffic-crash-death-estimates-2022.

[2] L. Chen, Y. Li, C. Huang, *et al.*, "Milestones in autonomous driving and intelligent vehicles: Survey of surveys," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 2, pp. 1046–1056, 2022.

[3] E. M. Abuhdima, A. E. Qaouaq, S. Alston, *et al.*, "Impact of weather conditions on 5G communication channel under connected vehicles framework," *arXiv:2111.09418 [cs.NI]*, 2021.

[4] M. N. Ahangar, Q. Z. Ahmed, F. A. Khan, and M. Hafeez, "A survey of autonomous vehicles: Enabling communication technologies and challenges," *Sensors*, vol. 21, no. 3, p. 706, 2021.

[5] S. Joerer, M. Segata, B. Bloessl, R. L. Cigno, C. Sommer, and F. Dressler, "A vehicular networking perspective on estimating vehicle collision probability at intersections," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 4, pp. 1802–1812, 2013.

[6] L. Tao, Y. Watanabe, Y. Li, S. Yamada, and H. Takada, "Collision risk assessment service for connected vehicles: Leveraging vehicular state and motion uncertainties," *IEEE Internet of Things Journal*, vol. 8, no. 14, pp. 11 548–11 560, 2021.

[7] Y. Fu, C. Li, T. H. Luan, Y. Zhang, and G. Mao, "Infrastructure-cooperative algorithm for effective intersection collision avoidance," *Transportation Research Part C: Emerging Technologies*, vol. 89, pp. 188–204, 2018.

[8] C. Chen, L. Liu, T. Qiu, Z. Ren, J. Hu, and F. Ti, "Driver's intention identification and risk evaluation at intersections in the Internet of vehicles," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1575–1587, 2018.

[9] M. Baek, D. Jeong, D. Choi, and S. Lee, "Vehicle trajectory prediction and collision warning via fusion of multisensors and wireless vehicular communications," *Sensors*, vol. 20, no. 1, p. 288, 2020.

[10] S. Y. Gelbal, S. Zhu, G. A. Anantharaman, B. A. Guvenc, and L. Guvenc, "Cooperative collision avoidance in a connected vehicle environment," *arXiv:2306.01889 [cs.RO]*, 2023.

[11] G. Avino, P. Bande, P. A. Frangoudis, *et al.*, "A MEC-based extended virtual sensing for automotive services," *IEEE Trans. on Network and Service Management*, vol. 16, no. 4, pp. 1450–1463, 2019.

[12] D. C. Selvaraj, C. Vitale, T. Panayiotou, P. Kolios, C. F. Chiasserini, and G. Ellinas, "Edge learning of vehicular trajectories at regulated intersections," in *Proc. IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*, 2021, pp. 1–7.

[13] D.C. Selvaraj, C. Vitale, T. Panayiotou, P. Kolios, C.F. Chiasserini, and G. Ellinas, "Edge-assisted ML-aided uncertainty-aware vehicle collision avoidance at urban intersections," *IEEE Transactions on Intelligent Vehicles*, pp. 1–1, 2023.

[14] Y. Huang, Y. Wang, X. Yan, X. Li, K. Duan, and Q. Xue, "Using a V2V-and V2I-based collision warning system to improve vehicle interaction at unsignalized intersections," *Journal of Safety Research*, vol. 83, pp. 282–293, 2022.

[15] Y. Fu, C. Li, F. R. Yu, T. H. Luan, and Y. Zhang, "A survey of driving safety with sensing, vehicular communications, and artificial intelligence-based collision avoidance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 6142–6163, 2021.

[16] W. Q. Meeker, G. J. Hahn, and L. A. Escobar, *Statistical Intervals: A Guide for Practitioners and Researchers*. Wiley, 2017.

[17] F. Rodrigues and F. Pereira, "Beyond expectation: Deep joint mean and quantile regression for spatiotemporal problems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 5377–5389, 2020.

[18] G. Louppe, "Understanding random forests: From theory to practice," *arXiv:1407.7502 [stat.ML]*, 2015.

[19] F. Kaltenberger, A. P. Silva, A. Gosain, L. Wang, and T.-T. Nguyen, "OpenAirInterface: Democratizing innovation in the 5G era," *Computer Networks*, vol. 176, p. 107 284, 2020.

[20] L. Abeni and D. Faggioli, "Using Xen and KVM as real-time hypervisors," *Journal of Systems Architecture*, vol. 106, p. 101 709, 2020.

[21] *ASN1 Compiler*, https://www.oss.com/asn1/products/asn1-cpp/asn1-cpp.html, 2024.

[22] *ONNX representation*, https://github.com/onnx/onnx/releases/tag/v1.15.0, 2024.

[23] L. Codecá, R. Frank, S. Faye, and T. Engel, "Luxembourg SUMO traffic (LuST) scenario: Traffic demand evaluation," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 2, pp. 52–63, 2017.

[24] S. Rezaei and R. Sengupta, "Kalman filter-based integration of dgps and vehicle sensors for localization," *IEEE Transactions on Control Systems Technology*, vol. 15, no. 6, pp. 1080–1088, 2007.